Stelian Ciurea
Adriana Simulescu
Carmen Popescu
Carmen Mincă
Cristina Sichim
Cristina Iordaiche

INFORMATICĂ

Culegere de probleme

pentru

Pregătirea bacalaureatului

L&S SOFT

Copyright 2019 © L&S SOFT

Toate drepturile asupra acestei lucrări aparţin editurii L&S SOFT. Reproducerea integrală sau parţială a textului din această carte este posibilă doar cu acordul în scris al editurii L&S SOFT.

Descrierea CIP a Bibliotecii Naţionale a României

Informatică : culegere de probleme pentru clasa a IX-XI şi pregătirea bacalaureatului / Stelian Ciurea, Carmen Mincă, Adriana Simulescu, ... - Bucureşti : Editura L&S Soft, 2007
ISBN 978-973-88037-3-2

I. Ciurea, Stelian
II. Mincă, Carmen
III. Simulescu, Adriana

004(075.35)
007(075.35)

ATENŢIE!

După confirmarea plății, fiecare carte poate fi descărcată de **maximum 5 ori** şi este **disponibilă 30 de zile**.

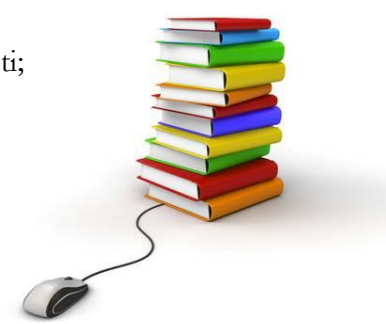
Fiecare PDF este **securizat în 28 de zone cu watermark invizibil** (id comandă, e-mail) pentru a **nu putea fi distribuit pe alte căi virtuale**.

Editura L&S SOFT

Adresa: Str. Stânjeneilor nr. 6, sector 4, Bucureşti;
Mobil: 0727.731.947, 0749-99.77.07;
E-mail: comenzi@ls-infomat.ro;

www.ls-infomat.ro
www.manuale-de-informatica.ro

Biblioteca Digitală de Informatică "Tudor Sorin"
www.infobits.ro



1. Algoritmi elementari şi date din tipurile simple

1.1. Teste grilă limbaj pseudocod

1. Se consideră primii **6** termeni din şirul Fibonacci: **1,1,2,3,5,8.** Precizaţi care dintre secvenţele de numere următoare **nu** reprezintă trei termeni succesivi din şirul Fibonacci:

1) **144,233,376** 2) **34,55,89** 3) **89,144,233**

a. numai **3** **b.** atât **2** cât şi **3** **c.** numai **1** **d.** numai **2**

2. Ce valoare va reţine variabila **x** după executarea următoarei secvenţe de atribuiri ?

x←10; y←3; x←x-y; y←x+y; x←y-x

a. **10** b. **-3** c. **7** d. **3**

3. Stabiliţi care dintre următoarele expresii logice are valoarea **ADEVĂRAT** dacă şi numai dacă valoarea variabilei reale **x** se găseşte în afara intervalului **[0,1]**.

a. **x<0 şi x>1**
b. **x<0 sau x>1**
c. **x≥0 şi x≤1**
d. **x≤0 sau x≥1**

4. Condiţia ca două numere întregi **a** şi **b** să fie ambele nenule este:

a. **(a≠0) sau (b≠0)**
b. **a*b≠0**
c. **a+b≠0**
d. **not ((a=0) şi (b=0))**

5. Ştiind că două intervale de numere reale **[a,b]** şi **[c,d]** îndeplinesc condiţia ca maximul dintre **a** si **c** este mai mic sau egal decât minimul dintre **b** si **d,** atunci intervalul **[max{a,c}, min{b,d}]** reprezintă:

a. **[a,b] ∩ [c,d]**
b. **[a,b] U [c,d]**
c. **[c,d] - [a,b]**
d. **[a,b] - [c,d]**

6. Care dintre secvenţele următoare afişează corect prima zecimală a numărului real pozitiv reţinut de **a**? S-a notat cu **[a]** partea întreagă a numărului real **a** şi cu **|a|** valoarea absolută a numărului real **a**.

a. **a←|a-[a]|; scrie [a*10]**
b. **a← [a-|a|]; scrie [a*10]**
c. **a←[a-|a|]; scrie [a]*10**
d. **a←|a-[a]|; scrie [a]*10**

7. Care din următoarele secvenţe de instrucţiuni în pseudocod testează dacă z este cea mai mare valoare dintre valorile variabilelor **x**, **y, z** şi în caz afirmativ o afişează?

a. dacă **(x<y)** şi **(y<z)** atunci scrie **z**

b. dacă **(x<y)** şi **(x<z)** atunci scrie **z**

c. dacă **(x<y)** sau (**y<z**) sau **(x<z)** atunci scrie **z**

d. dacă **(x>z)** sau **(y>z)** atunci scrie **z**

8. Ştiind că variabilele **a** şi **i** sunt întregi, stabiliţi ce reprezintă valorile afişate de algoritmul alăturat.
S-au folosit notaţiile **x%y** pentru restul împărţirii numărului întreg **x** la numărul întreg **y**, şi **[x]** pentru partea întreagă a numărului real **x**.

```
a←10
pentru i=1,6 execută
  scrie [a/7]
  a←a%7*10
sfârşit pentru
```

a. primele **6** zecimale ale lui **1/7**

b. primele **7** zecimale ale lui **1/6**

c. primele **6** zecimale ale lui **10/7**

d. primele **7** zecimale ale lui **10/6**

9. Se consideră algoritmul pseudocod alăturat scris neindentat. Stabiliţi care este numărul maxim de structuri de decizie imbricate (conţinute una în alta) din acest algoritm.

```
dacă a<=b atunci dacă a=b atunci scrie "egale" altfel scrie "primul" sfârşit dacă altfel scrie "al doilea" sfârşit dacă
dacă a+b<100 atunci scrie "mici" altfel scrie "mari" sfârşit dacă
```

a. 3 **b.** 1 **c.** 2 **d.** 4

10. Care sunt valorile variabilelor întregi **a** şi **b** după executarea secvenţei alăturate, dacă iniţial ele aveau valori diferite?

```
┌dacă (a<b) sau (a>b) atunci
│ a ← 1; b ← a
└■
┌dacă a=b atunci
│ b ← 0
└■
```

a. **a=1 şi b=1**

b. **a=1 şi b=0**

c. **a=0 şi b=0**

d. **a=0 şi b=1**

11. Ce va afişa algoritmul pseudocod alăturat pentru două numere naturale nenule **a** şi **b**?
S-a notat cu **x%y** restul împărţirii numerelor întregi **x** şi **y**.

```
citeşte a,b (numere naturale)
c ← 1
┌cât timp a*c%b≠0 execută
│   c ← c + 1
└■
scrie a*c
```

a. a^b

b. cel mai mic multiplu comun

c. cel mai mare divizor comun

d. **a*b**

12. Ce reprezintă rezultatul afişat de programul pseudocod alăturat?

```
citeşte n (n>0 nr natural)
s←0; k←1
┌cât timp k<=n execută
│ s←s+k; k←k+2
└■
scrie s
```

a. suma numerelor naturale impare mai mici sau egale decât **n**.

b. suma primelor **n** numere naturale.

c. suma numerelor naturale impare mai mici decât **n**.
d. suma numerelor naturale pare mai mici decât **n**.

13. Precizaţi valoarea afişată de algoritmul pseudocod alăturat, dacă s-a notat cu **x%y** restul împărţirii lui **x** la **y**, iar cu **[x]** partea întreagă a numărului real **x**.

```
a←12345; s←0
┌cât timp a>0 execută
│ s←s+a%2; a←[a/10]
└■
scrie s
```

a. 3 **b.** 5 **c.** 15 **d.** 9

14. Fie algoritmul pseudocod alăturat. Care dintre următorii algoritmi este echivalent cu algoritmul dat?
S-a notat cu **[x]** partea întreagă a numărului **x** şi cu **x%y** restul împărţirii lui **x** la **y**.

```
p ← 0
citeşte a, b
      (numere naturale)
┌cât timp a>=b execută
│     p←p+1; a←a-b
└■
scrie p
```

a. `citeşte a, b` (numere naturale)
`p←b/a; scrie p`

b. `citeşte a, b` (numere naturale)
`p←a % b; scrie p`

c. `citeşte a, b` (numere naturale)
`p←[a/b]; scrie p`

d. `citeşte a, b` (numere naturale)
`p←a * b; scrie p`

15. Cerinţa: "citeşte mai multe numere naturale pâna la întâlnirea numărului 0" poate fi implementată prin:

a. O structură repetitivă cu număr necunoscut de paşi
b. O structură alternativă
c. O structură repetitivă cu număr cunoscut de paşi
d. Nu se poate implementa

16. Fie algoritmul pseudocod alăturat. Ce afişează algoritmul dacă pentru **a** se introduc pe rând numerele **2, -2.5, 4, 8, 0**?

```
p ← 1
citeşte a (număr real)
┌cât timp a≠0 execută
│  p ← p*a; citeşte a (număr real)
└■
scrie p
```

a. -160 **b.** 160 **c.** 0 **d.** 1

17. În secvenţa pseudocod alăturată, variabilele **i** şi **j** sunt de tip întreg. Care dintre următoarele valori poate fi valoarea iniţială a variabilei **i** astfel încât secvenţa să scrie un număr finit de valori?

```
 j←1
┌cât timp i≥j execută
│ i←i+1; scrie i
└■
 scrie i
```

a. 10 **b.** 1 **c.** 100 **d.** 0

18. Pentru un număr natural memorat în variabila **a**, stabiliţi ce reprezintă rezultatul afişat de programul pseudocod alăturat.

```
┌cât timp a>9 execută
│   a←a-10
└■
 scrie a
```

a. ultima cifră a numărului **a**
b. prima cifră a numărului **a**
c. valoarea obţinută prin eliminarea ultimei cifre a lui **a**
d. numărul de cifre ale lui **a**

19. Pentru o valoare reală mai mare decât **1** memorată în variabila **a**, rezultatul afişat de programul pseudocod alăturat reprezintă valoarea întreagă cea mai apropiată de **a** dintre numerele:

```
i←1
┌cât timp i*a<a^2 execută
│    i←i+1
└■
scrie i
```

a. strict mai mari decât **a**
b. mai mici sau egale cu **a**
c. strict mai mici decât **a**
d. mai mari sau egale cu **a**

20. Care este valoarea maximă pe care o poate afişa algoritmul alăturat?

S-a notat cu **x%y** restul împărţirii numerelor întregi **x** şi **y**.

```
citeşte a (număr natural)
i←0; a←a%10
┌cât timp (a>1)şi(a<10) execută
│i←i+1; a←a*a
└■
scrie i*a
```

a. 243 b. 32 c. 81 d. 162

21. Se consideră secvenţa de instrucţiuni în pseudocod alăturată. Ce valoare trebuie scrisă în pătrăţel pentru a se afişa **165**?

a. 8 **b.** 9
c. 10 **d.** 12

```
i←3;E←1
┌cât timp i< [ ] execută
│ ┌dacă i mod 2≠0 atunci
│ │  E←E+i*i
│ └■
│ i←i+1
└■
scrie E
```

22. Cu ce expresie trebuie înlocuite punctele de suspensie astfel încât programul pseudocod alăturat să memoreze în variabila **p** produsul celor **10** numere întregi negative citite?

```
p←…
┌pentru i=1,10 execută
│ citeşte x (număr întreg, x<0)
│ p←p*x
└■
scrie p
```

a. 0 **b.** 10 **c.** 1 **d.** -1

23. Algoritmul alăturat atribuie variabilei **min** cea mai mică valoare întreagă dintre **n** numere întregi, mai mici decât **100**, citite de la tastatură.

Care dintre valorile de mai jos poate înlocui punctele de suspensie astfel încât algoritmul să furnizeze rezultatul corect?

```
citeste n (număr natural, n<50)
min←...
┌pentru i←1,n executa
│  citeste x (număr întreg x<100)
│  daca x<min atunci min←x
└■
scrie min
```

a. 100 **b.** 1 **c.** 0 **d.** -100

24. De câte ori se execută instrucţiunea **scrie x** din programul pseudocod următor?

```
x←5
┌ pentru i←5,8 execută
│   ┌ pentru j←i-x,i execută
│   │   ┌ dacă (x>3) and (i>8) and (j<7)atunci scrie x
│   │   └■
│   └■
└■
```

a. 18 b. 8 c. 15 d. 0

25. Precizaţi care este valoarea afişată de algoritmul pseudocod alăturat?

```
y←0; x←0
┌pentru i←-1,3 execută
│  x←y+i; y←x
└■
scrie x
```

a. 0 **b.** 3 **c.** 6 **d.** 5

26. Secvenţa alăturată afişează:

```
citeşte S    (număr natural, S<=18)
┌pentru i=1,9 execută
│  ┌dacă (S-i)≥0 şi (S-i)≤9 atunci
│  │  afişează 10*i+(S-i)
│  └■
└■
```

a. numere naturale cu cifre distincte, mai mici decât S
b. numere naturale mai mari decât 10 şi mai mici decât S
c. numerele naturale care au suma cifrelor egală cu S
d. numerele naturale de două cifre care au suma cifrelor egală cu S

27. Cu ce expresie trebuie înlocuite punctele de suspensie astfel încât programul pseudocod alăturat să memoreze în variabila **p** produsul celor **10** numere întregi negative citite?

```
p←…
┌pentru i=1,10 execută
│ citeşte x (număr întreg, x<0)
│ p←p*x
└■
scrie p
```

a. 0 **b.** 10 **c.** 1 **d.** -1

28. Cu ce expresie trebuie înlocuite punctele de suspensie astfel încât algoritmul pseudocod alăturat să calculeze şi să afişeze cea mai mică cifră din scrierea în baza **10** a unui număr natural **n**?
S-a folosit notaţia **[x]** pentru partea întreagă a lui **x**.

```
citeşte n (număr natural)
m←…
┌repetă
│ ┌dacă n%10<m atunci
│ │ m←n%10
│ └■
│  n←[n/10]
└până când n=0
scrie m
```

a. `100` **b.** `0` **c.** `[n/10]` **d.** `1`

29. Pentru **a** şi **b** numere naturale cunoscute, secvenţa următoare afişează mesajul **DA** numai dacă numărul **a** este un prefix al lui **b** şi afişează **NU** în caz contrar. De exemplu, pentru **a=73** şi **b=7306**, afişează **DA**. Pentru **a=8** şi **b=508**, sau **a=61** şi **b=61**, sau **a=0** şi **b=33**, afişează **NU**. S-a folosit notaţia **[x]** pentru partea întreagă a numărului real pozitiv **x**. Care este expresia cu care trebuie completate punctele de suspensie?

```
repetă
   b←[b/10]
până când b=0 sau a=b
dacă .... atunci scrie DA
   altfel scrie NU
sfârşit dacă
```

a. `a≠b`
b. `a=b`
c. `b=0`
d. `b≠0`

30. Cu ce expresie trebuie înlocuite punctele de suspensie astfel încât algoritmul pseudocod alăturat să calculeze şi să afişeze cea mai mică cifră din scrierea în baza **10** a unui număr natural **n**?
S-a folosit notaţia **[x]** pentru partea întreagă a lui **x**.

```
citeşte n (număr natural)
m←…
┌repetă
│ ┌dacă n%10<m atunci
│ │ m←n%10
│ └■
│  n←[n/10]
└până când n=0
scrie m
```

a. `100` **b.** `0` **c.** `[n/10]` **d.** `1`

Teste grilă - limbaj de programare

(elemente de bază, Pascal | C / C++)

1. Ştiind că variabila **x** este utilizată pentru a memora media geometrică a **2** numere naturale cu cel mult patru cifre, stabiliţi care este declararea corectă a variabilei **x.**

a. `var x:[1..2,1..4] of word;`	**a.** `unsigned int x;`
b. `var x:array[1..2,1..4] of real;`	**b.** `float x[2][4];`
c. `var x:real;`	**c.** `double x;`
d. `var x:array[1..2,1..4]of integer;`	**d.** `int x[2][4];`

2. Dintre tipurile simple de date face parte tipul:

a. `real`	**c.** `pointer`	**a.** `float`	**c.** `pointer`
b. `fişier`	**d.** `tablou`	**b.** `fişier`	**d.** `tablou`

3. Variabila **x** este utilizată într-un program pentru a memora rezultatul **1*2*3*…*10.** Stabiliţi care este declararea corectă pentru variabila **x:**

a. `var x:byte;`	**a.** `char x;`
b. `var x:integer;`	**b.** `int x;`
c. `var x:longint;`	**c.** `long x;`
d. `var x:string[5];`	**d.** `char x[5];`

4. Dacă **x** este o variabilă care memorează şi utilizează în calcule rezultatul împărţirii a două numere întregi, cu două zecimale exacte, care dintre următoarele declaraţii ale variabilei **x** este corectă?

a. `var x : array[1..9,1..2] of integer;`	**a.** `int x[9][2];`
b. `var x : word;`	**b.** `unsigned int x;`
c. `var x : string[2];`	**c.** `char x[2];`
d. `var x : real;`	**d.** `float x;`

5. Care dintre următoarele variante **nu** reprezintă o declarare corectă a unei variabile de tip caracter?

a. `Var ab: char;`	**a.** `char ab;`
b. `Var a&b: char;`	**b.** `char a&b;`
c. `Var a_b: char;`	**c.** `char a_b;`
d. `Var a1b: char;`	**d.** `char a1b;`

6. Ştiind că variabila **x** este utilizată pentru a memora produsul a **12** numere reale (cu cel mult trei zecimale), stabiliţi care este declararea corectă a variabilei **x.**

a. `var x: array[1..12] of real;`	**a.** `float x[12];`
b. `var x: array[1..12,1..3] of real;`	**b.** `float x[12][3];`
c. `var x: real;`	**c.** `float x;`
d. `var x: array[1..3]of real;`	**d.** `float x[3];`

7. Ştiind că variabila **g** este utilizată pentru a memora şi utiliza în calcule valoarea acceleraţiei gravitaţionale (cu două zecimale), atunci declararea corectă a variabilei **g** este:

a. `Var g: array[1..2]of byte;`	**a.** `long g[2];`
b. `Var g:array[1..2]of integer;`	**b.** `int g;`
c. `Var g:real;`	**c.** `float g;`
d. `Var g:integer;`	**d.** `int g[2];`

8. Care dintre următoarele instrucţiuni iniţializează variabila intreagă **n** cu valoarea **2**? Se presupune că iniţializarea unei variabile este instrucţiunea prin care se atribuie o valoare în momentul declarării

a. `var n:=2`	**a.** `int n:=2`
b. `const n:integer=2`	**b.** `int n=2`
c. `var n=2`	**c.** `int n==2`
d. `int n=2`	**d.** `int n`

9. Care dintre următoarele afirmaţii este adevărată?

a. În limbajul Pascal/C/C++ nu există noţiunea de operator relaţional
b. `+, -, *, /` sunt operatori relaţionali
c. `<,>, <=, >=` sunt operatori relaţionali
d. `&&, ||, !` sunt operatori relaţionali

10. Lungimea cercului este produsul dintre π şi diametrul cercului, iar diametrul este dublul razei cercului. Ştiind că variabila reală **R** memorează o valoare strict pozitivă reprezentând raza cercului, iar variabila reală **P** memorează cu suficientă precizie valoarea numărului π, precizaţi care dintre următoarele expresii scrise în Pascal/C/C++ are ca valoare lungimea cercului.

a. `2*π*R` **b.** `2R*P` **c.** `2*P*R` **d.** `P*R/2`

11. Precizaţi valoarea expresiei: `8/4/2*2*4*8.`

a. `64` **b.** `1` **c.** `16` **d.** `0`

12. Care este valoarea expresiei următoare?

`(2*3-5 div 2 -(3-7 mod 2))`	`(2*3-5/2-(3-7%2))`

a. `-2` **b.** `1.5` **c.** `-1.5` **d.** `2`

13. Cea mai mică valoare strict pozitivă pe care o poate avea variabila întreagă **x** pentru ca expresia `(x div 2)*2 | (x/2)*2` să fie egală cu **x** este:

a. `1` **b.** `3` **c.** `2` **d.** `0`

14. Care dintre următoarele expresii logice **nu** este echivalentă cu condiţia ca valoarea variabilei întregi pozitive **n** să fie formată dintr-o singură cifră?

a. `n=n mod 10`	**c.** `n<=n mod 10`	**a.** `n==n%10`	**c.** `n<=n%10`
b. `n div 10=0`	**d.** `n<=10`	**b.** `n/10==0`	**d.** `n<=10`

15. Care dintre următoarele expresii logice este echivalentă cu conditia ca variabilele întregi **a** şi **b** să aibă acelaşi semn şi să fie nenule?

a. **a*b>0**	**a.** **a*b>0**
b. **a+b>0**	**b.** **a+b>0**
c. `((a>0) or (b>0)) and ((a<0) or (b<0))`	**c.** `(a>0 || b>0) && (a<0 || b<0)`
d. `not ((a<0)and (b<0) and (a>0)and (b>0))`	**d.** `!(a<0 && b>0 || a>0 && b<0)`

16. Condiţia ca două variabile întregi **a** şi **b**, despre care se ştie că reţin valori pozitive, să aibă simultan valoarea zero este:

a. `not((a<>0) and (b<>0))`	**a.** `!(a && b)`
b. `a*b=0`	**b.** `a*b==0`
c. `a+b=0`	**c.** `a+b==0`
d. `(a=0) or (b=0)`	**d.** `a==0||b==0`

17. Care dintre următoarele expresii logice este echivalentă cu conditia ca variabilele întregi pozitive **a** şi **b** să fie ambele pare sau ambele impare?

a. `(a+b) mod 2=0`	**a.** `(a+b)%2==0`
b. `(a mod 2=0) or (b mod 2=0)`	**b.** `a%2==0||b%2==0`
c. `(a mod 2<>0) and (b mod 2<>0)`	**c.** `a%2 && b%2`
d. `not (a*b mod 2<>0)`	**d.** `!(a*b%2)`

18. Fie **a**, **b** şi **c** oricare trei numere naturale nenule. Ştiind că **a** este divizor al lui **b**, iar **c** nu este multiplu al lui **a**, care dintre următoarele expresii are valoare **1** în **C/C++**, respectiv **true** în **Pascal**?

a. `not((b mod a <> 0) or (c mod a =0))`	**a.** `!((b % a) || !(c % a))`
b. `(b mod a <> 0) or (c mod a = 0)`	**b.** `(b % a) || !(c % a)`
c. `(a mod b = 0) and (a mod c <> 0)`	**c.** `!(a % b) && (a % c)`
d. `(b mod a <> 0) and (c mod a = 0)`	**d.** `(b % a) && !(c % a)`

19. Fie variabilele reale **a**, **b** şi **x** cu **a <= b**. Care dintre următoarele expresii logice este adevărată, daca şi numai dacă **x**$\notin$ **[a, b]** ?

a. `(x >= a) or (x <= b)`	**a.** `(x >= a) || (x <= b)`
b. `(x > a) and (x < b)`	**b.** `(x > a) && (x < b)`
c. `not (x >= a) or not (x <= b)`	**c.** `!(x >= a) || !(x <= b)`
d. `(x >= a)and (x <= b)`	**d.** `(x >= a) && (x <= b)`

20. Cum scriem în Pascal/C/C++ că **x∉[a,b]**?

a. `(x<a) and (x>b)`	**a.** `x<a && x>b`
b. `(x<=a) or (x>=b)`	**b.** `x<=a || x>=b`
c. `(x<a) or (x>b)`	**c.** `x<a || x>b`
d. `not((a<=x) or (x<=b))`	**d.** `!(a<=x || x<=b)`

21. Valoarea variabilei **x** este un număr real. Pentru a verifica apartenenţa valorii variabilei **x** la mulţimea de valori **[-2,2]U{3, 4, 7}** se va utiliza următoarea expresie:

Varianta Pascal	**a.**	`not ((x<-2) or (x>2)) or (x=3) or (x=4) or (x=7)`
	b.	`(x>=-2) and (x<=2) and (x=3) or (x=4) or (x=7)`
	c.	`(x>-2) and (x<2) or (x=3) or (x=4) or (x=7)`
	d.	`(x<-2) or (x>2) and (x=3) and (x=4) and (x=7)`
Varianta C/C++	**a.**	`!((x<-2)||(x>2)) || (x==3)||(x==4)||(x==7)`
	b.	`(x>=-2)&&(x<=2)&&(x==3)||(x==4)||(x==7)`
	c.	`(x>-2)&&(x<2)||(x==3)||(x==4)||(x==7)`
	d.	`(x<-2)||(x>2)&&(x==3)&&(x==4)&&(x==7)`

22. Dacă expresia de mai jos:

`(x>-2)and(not(x>2))or(x>=5)`	`(x>-2)&&(!(x>2))||(x>=5)`

este adevărată, atunci:

a.	**x**∈ `(-2,2]` ∩ `[5,∞)`	**c.**	**x**∈ `(-2,2)` **U** `[5,∞)`
b.	**x**∈ `(-2,2)` ∩ `(5,∞)`	**d.**	**x**∈ `(-2,2]` **U** `[5,∞)`

23. Stabiliţi care dintre următoarele expresii este adevărată dacă şi numai dacă **x** este in afara intervalului închis **[a,b]**

a. `(x<a) and (x>b)`	**a.** `x<a && x>b`
b. `(x>=a) and (x<=b)`	**b.** `x>=a && x<=b`
c. `not((x>=a) and (x<=b))`	**c.** `!(x>=a && x<=b)`
d. `not((x<a) or (x>b))`	**d.** `!(x<a || x>b)`

24. Care din următoarele expresii este adevărată dacă şi numai dacă numărul întreg **a** este un număr impar pozitiv?

a. `(a mod 2=0) and (a>0)`	**a.** `a%2==0&&a>0`
b. `not ((a mod 2=0) and (a<=0))`	**b.** `!(a%2==0&&a<=0)`
c. `(a>0)or(a mod 2=1)`	**c.** `a>0||a%2==1`
d. `not((a<=0)or(a mod 2=0))`	**d.** `!(a<=0||a%2==0)`

25. Ştiind că variabila întreagă **a** reţine o valoare naturală de cel mult 3 cifre, stabiliţi care dintre următoarele expresii este adevărată dacă şi numai dacă **a** este format numai din cifre pare?

Varianta Pascal:	**a.**	`(a mod 2=0)and(a mod 10 mod 2=0)and(a mod 100 mod 2=0)`
	b.	`(a mod 2=0)and(a div 10 mod 2=0)and(a div 100 mod 2=0)`
	c.	`(a div 10 mod 2=0)and(a div 100 mod 2=0)`
	d.	`(a div 2=0)and(a mod 10 mod 2=0)and(a mod 100 mod 2=0)`
Varianta C/C++	**a.**	`a%2==0 && a%10%2==0 && a%100%2==0`
	b.	`a%2==0 && a/10%2==0 && a/100%2==0`
	c.	`a/10%2==0 && a/100%2==0`
	d.	`a/2==0 && a%10%2==0 && a%100%2==0`

26. Valoarea memorată în variabila reală **x** aparţine intervalului închis **[a,b]** cu **a<b**. Care dintre următoarele expresii logice are valoarea **true/1**?

a.	`not((x<a) or (x>b))`	**a.**	`!((x<a) || (x>b))`
b.	`(a<x) and (b>x)`	**b.**	`(a<x) && (b>x)`
c.	`not((x<=a) or (x>=b))`	**c.**	`!((x<=a) || (x>=b))`
d.	`(x<=a) and (x>=b)`	**d.**	`(x<=a) && (x>=b)`

27. Care dintre următoarele expresii are valoarea **TRUE/1** dacă şi numai dacă numărul natural **c** este un multiplu comun al numerelor naturale nenule **a** şi **b**?

a.	`(c mod a=0) or (c mod b=0)`	**a.**	`(c%a==0) || (c%b==0)`
b.	`(a mod c=0) or (b mod c=0)`	**b.**	`(a%c==0) || (b%c==0)`
c.	`(a mod b=0) and (c mod b=0)`	**c.**	`(a%b==0) && (c%b==0)`
d.	`(c mod a=0) and (c mod b=0)`	**d.**	`(c%a==0) && (c%b==0)`

28. Ştiind că variabilele **a** şi **b** de tip întreg memorează două numere naturale pare, care dintre următoarele expresii are valoarea **true/1**?

a.	`not((a mod 2=0) and (b mod 2=0))`	**a.**	`a%2 && b%2`
b.	`not(not(a mod 2=1) and not(b mod 2=1))`	**b.**	`!(!(a%2) || !(b%2))`
c.	`not((a mod 2=1) or (b mod 2=1))`	**c.**	`!(a%2 || b%2)`
d.	`not((a+b) mod 2=0)`	**d.**	`(a+b)%2`

29. Ştiind că variabilele **a** şi **b** de tip întreg memorează două numere naturale impare, care dintre următoarele expresii are valoarea **true/1**?

a.	`(a mod 2=0) or (b mod 2=0)`	**a.**	`!(a%2) || !(b%2)`
b.	`not((a mod 2=1) or (b mod 2=1))`	**b.**	`!(a%2 || b%2)`
c.	`not(a mod 2=1) and not(b mod 2=1)`	**c.**	`!(a%2) && !(b%2)`
d.	`not(not(a mod 2=1) or not(b mod 2=1))`	**d.**	`!(!(a%2) || !(b%2))`

30. Care dintre următoarele expresii este echivalentă cu expresia următoare?

`not(((a<=b)and(a>=c))or(a<=d))`	`!((a<=b && a>=c) || a<=d)`
a. `(a<=b) or (a>=c) and (a<=d)`	**a.** `a<=b || a>=c && a<=d`
b. `(a>b) or (a<c) and (a>d)`	**b.** `a>b || a<c && a>d`
c. `(a>b) and (a<c) or (a>d)`	**c.** `a>b && a<c || a>d`
d. `((a>b) or (a<c)) and (a>d)`	**d.** `(a>b || a<c) && a>d`

31. Ştiind că variabila reală **x** are o valoare ce aparţine intervalului închis `[3,7]`, care dintre expresiile de mai jos, **NU** are valoarea de adevăr **`true/1`** ?

a. `not((x<3) or (x>7))`	**a.** `!(x<3 || x>7)`
b. `not(x<3) and not (x>7)`	**b.** `!(x<3) && ! (x>7)`
c. `(x>=3) and (x<=7)`	**c.** `x>=3 && x<=7`
d. `(3<x) and (x<=7)`	**d.** `3<x && x<=7`

32. Stabiliţi care dintre următoarele expresii este adevărată dacă şi numai dacă **`numărul întreg x, nu`** aparţine intervalului închis `[20,100]`.

a. `(x<=20)or(x>=100)`	**a.** `(x<=20)||(x>=100)`
b. `(x<20)or not(x>100)`	**b.** `(x<20)|| !(x>100)`
c. `(x<=19)or(x>=101)`	**c.** `(x<=19)||(x>=101)`
d. `(x<=19)and(x>=101)`	**d.** `(x<=19)&&(x>=101)`

33. Valoarea expresiei următoare :

Varianta Pascal	`(x>-2)and(not(x>2))or(x>=5)and(x<10)`
Varianta C/C++	`(x>-2)&&(!(x>2))||(x>=5)&&(x<10)`

este **`TRUE/1`** dacă şi numai dacă **`x`** aparţine intervalului

a. `(-2,2]∪[5,10)` **b.** `[-2,2] ∪[5,10)` **c.** `(2,5]` **d.** `(-2,10)`

34. Valoarea expresiei următoare :

Varianta Pascal	`(x>-3)and(not(x>3))or (x>=5)and (x<10)`
Varianta C/C++	`(x>-3)&&(!(x>3))||(x>=5) &&(x<10)`

este **`TRUE/1`** dacă şi numai dacă **`x`** aparţine intervalului

a. `(-3,10)` **b.** `(3,5]` **c.** `[-3,3]∪[5,10)` **d.** `(-3,3]∪[5,10)`

35. Care dintre următoarele expresii are valoarea **TRUE (1** pentru C/C++) ştiind că variabilele **a,b,c** şi **d** de tip întreg au valorile **a=1, b=2, c=3, d=2** ?

a. `(a=c) and (b or d)`	**a.** `(a==c) && (b||d)`
b. `(b>c) or (c>3)`	**b.** `(b>c) || (c>3)`
c. `((b=d)and(a<>0)) or (b<=c)`	**c.** `((b==d)&&(a!=0))||(b<=c)`
d. `(b>c) and a`	**d.** `(b>c) && a`

36. Dacă **x, a** şi **b** reprezintă variabilele reale şi **a<b**, ce expresie se utilizează într-un program pentru a testa dacă valoarea variabilei **x** este situată în intervalul închis **[a,b]** ?

a. `a<=x<=b`	**a.** `a<=x<=b`
b. `(x>=a) and (x<=b)`	**b.** `(x>=a) && (x<=b)`
c. `(x>a) and (x<=b)`	**c.** `(x>a) && (x<=b)`
d. `(x>=a) or (x<=b)`	**d.** `(x>=a) || (x<=b)`

37. Care dintre expresiile următoare, scrise în limbajul Pascal|C/C++, are valoarea **true/1**?

a. `'a'='A'` **b.** `'1'+'2'='3'`	**a.** `'a'=='A'` **b.** `'1'+'2'=='3'`
c. `'a'<'b'` **d.** `'1'>'2'`	**c.** `'a'<'b'` **d.** `'1'>'2'`

38. Ce valoare are variabila întreagă **n** în urma executării secvenţei următoare?

```
n:=0;
repeat
   n:=n+3
until n>10;
```

```
n=0;
do
   n=n+3;
while(n<=10);
```

a. 10 **b.** 12 **c.** 11 **d.** 9

39. Ştiind că variabilele **a, b** şi **c** sunt de tip întreg, care este condiţia ca numărul natural memorat de variabila **c** să fie un multiplu comun al numerelor naturale memorate de variabilele **a** şi **b** ?

a. `c mod a * c mod b = 0`	**a.** `c%a * c%b == 0`
b. `a mod c + b mod c = 0`	**b.** `a%c + b%c == 0`
c. `c mod a + c mod b = 0`	**c.** `c%a + c%b == 0`
d. `(a mod c = 0)and(c mod b = 0)`	**d.** `(a%c == 0)&&(c%b == 0)`

40. Un an este bisect dacă este multiplu de **400** sau dacă este multiplu de **4** şi **nu** este multiplu de **100.** Care dintre următoarele expresii are valoarea **TRUE/1** dacă valoarea memorată de variabila **y** de tip întreg reprezintă un an bisect ?

a.	`y mod 4=0`	**a.**	`y%4==0`
b.	`(y mod 4=0) and (y mod 100=0) or (y mod 100<>0)`	**b.**	`(y%4==0)&&(y%100==0) || (y%100!=0)`
c.	`(y mod 4=0) and (y mod 100<>0) or (y mod 400=0)`	**c.**	`(y%4==0)&&(y%100!=0) || (y%400==0)`
d.	`(y div 400=0) or (y div 4=0) and (y div 100<>0)`	**d.**	`(y/400==0)||(y/4==0) && (y/100!=0)`

41. Fie variabilele întregi **x**, **a** şi **b**. Stabiliţi care dintre următoarele expresii este adevărată, dacă şi numai dacă ***x*** aparţine intervalului deschis **(a,b)**:

a.	`not ((x <= a ) or (x>=b))`	**a.**	`!((x <= a )|| (x >= b))`
b.	`(x >= a) and (x <= b)`	**b.**	`(x >= a) && (x <= b)`
c.	`(x >= a) or (x <= b )`	**c.**	`(x >= a) || (x <= b )`
d.	`not ((x <= a) and (x>=b))`	**d.**	`!((x <= a) && (x >= b))`

42. Care dintre următoarele expresii testează dacă un număr natural, strict pozitiv **n**, are ultima cifră egală cu **0** sau **5**?

a.	`(n>0) and (not (n mod 10<>0) or not ( n mod 5 <> 0))`	**a.**	`n>0&&(!(n%10)||!(n % 5))`
b.	`(n>0) and not (n mod 10<>0) and not (n mod 5 <> 0)`	**b.**	`n>0 && !(n%10) && !(n%5)`
c.	`(n > 0) and (n mod 10 <> 0) or (n mod 5 <> 0)`	**c.**	`n>0 && ((n % 10)||(n%5))`
d.	`(n > 0) and (n mod 10 <> 0) and (n mod 5 <> 0)`	**d.**	`n>0 && (n%10) && (n%5)`

43. Fie **a** o variabila întreagă, cu **a = 2006.** Care dintre următoarele expresii logice are valoarea **TRUE/1** ?

a.	`(a mod 2 div 10 = a div 10 mod 5) and (a mod 2 <> 0)`	**a.**	`(a%2/10==a/10%5)&&(a%2)`
b.	`(a mod 5 div 10 = a - a div 10 mod 2) and (a mod 5 <> 0)`	**b.**	`(a%5/10==a-a/10%2) && (a%5)`
c.	`(a mod 2 div 10 = a div 10 mod 2) and (a mod 5 <> 0)`	**c.**	`(a%2/10==a/10%2)&&(a%5)`
d.	`(a mod 2 div 10 = a div 10 mod 2) and (a mod 2 <> 0)`	**d.**	`(a%2/10==a/10%2)&&(a%2)`

44. Dintre trei valori reale memorate în variabilele **a**, **b**, **c**, oricare două sunt diferite dacă şi numai dacă:

a.	`(a<>b)and(a<>c)`	**a.**	`a!=b && a!=c`
b.	`(a<>b)and(b<>c)`	**b.**	`a!=b && b!=c`
c.	`(a<>b)and(a<>c)and(b<>c)`	**c.**	`a!=b && a!=c && b!=c`
d.	`(a<>b)or(a<>c)or(b<>c)`	**d.**	`a!=b || a!=c || b!=c`

45. Două valori naturale nenule memorate în variabilele **a** şi **b** au proprietatea că una divide pe cealaltă dacă şi numai dacă există un număr natural **c** astfel încât:

a. `(b*c=a)or(a*c=b)or(a*b=c)`	**a.** `(b*c==a)||(a*c==b)||(a*b==c)`
b. `(a=b*c)and(b=a*c)and(c=a*b)`	**b.** `(a==b*c)&&(b==a*c)&&(c==a*b)`
c. `(a=b*c)and(b=a*c)`	**c.** `(a==b*c)&&(b==a*c)`
d. `(a=b*c)or(a*c=b)`	**d.** `(a==b*c)||(a*c==b)`

46. Fie **a,b** numere reale cu **a<=b**. Numărul real **x** se găseşte în afara intervalului închis **[a,b]** dacă şi numai dacă:

a. `(x<=a)or (x>=b)`	**a.** `x<=a || x>=b`
b. `(x<a)or (x>b)`	**b.** `x<a || x>b`
c. `(x>=a)and (x<=b)`	**c.** `x>=a && x<=b`
d. `(x<a)and(x>b)`	**d.** `x<a && x>b`

47. Care dintre următoarele expresii sunt adevărate dacă şi numai dacă valorile variabilelor întregi **x** şi **y** au parităţi diferite?

Pascal	**a.**	`(x mod 2=0)and(x mod 2<>0)or(y mod 2=0) and (y mod 2<>0)`
	b.	`(x mod 2=0) or (x mod 2<>0)and(y mod 2=0) or (y mod 2<>0)`
	c.	`(x mod 2=0) or (y mod 2<>0)and(x mod 2<>0) or (y mod 2=0)`
	d.	`(x mod 2=0)and(y mod 2<>0) or (x mod 2<>0)and (y mod 2=0)`
C/C++	**a.**	`x%2==0 && x%2!=0 || y%2==0 && y%2!=0`
	b.	`x%2==0 || x%2!=0 && y%2==0 || y%2!=0`
	c.	`x%2==0 || y%2!=0 && x%2!=0 || y%2==0`
	d.	`x%2==0 && y%2!=0 || x%2!=0 && y%2==0`

48. Condiţia ca numărul natural **m** să fie multiplu al numerelor naturale **a** şi **b** este:

a. `(a mod m=0) or (b mod m=0)`	**a.** `a%m==0|| b%m==0`
b. `(a mod m=0) and (b mod m=0)`	**b.** `a%m==0 && b%m==0`
c. `(m mod a=0) or (m mod b=0)`	**c.** `m%a==0 || m%b==0`
d. `(m mod a=0) and (m mod b=0)`	**d.** `m%a==0 && m%b==0`

49. Niciunul dintre numerele reale **x** şi **y** **nu** aparţine intervalului **[a,b]** dacă şi numai dacă:

Varianta Pascal	**a.**	`((x<a) or (x>b)) and ((y<a) or (y>b))`
	b.	`(x<a) and (y<a) or (x>b) and (y>b)`
	c.	`(x<a) or (x>b) and (y<a) or (y>b)`
	d.	`(x*y<a*a) or (x*y>b*b)`
Varianta C/C++	**a.**	`(x<a || x>b) && (y<a || y>b)`
	b.	`x<a && y<a || x>b && y>b`
	c.	`x<a || x>b && y<a || y>b`
	d.	`x*y<a*a || x*y>b*b`

50. Pentru variabilele întregi **x,y,z** şi **t** ce memorează valorile **x=3, y=5, z=3, t=1**, precizaţi care dintre următoarele expresii logice are valoarea adevărat:

a. `(t<>0) and (x=z) or (y>=z)`	**a.** `(t!=0) && (x==z) || (y>=z)`
b. `(x>y) and (t<>0)`	**b.** `(x>y) && (t !=0)`
c. `(x=z) and (t=0)`	**c.** `(x==z) && (t==0)`
d. `not((x<>0) and (y<>0) and (z<>0))`	**d.** `!(x !=0 && y !=0 && z !=0)`

51. Dacă **a,b,z** sunt variabile reale şi **a≤b,** atunci care dintre expresiile următoare are valoarea adevărat dacă şi numai dacă **z∉[a,b]?**

a. `(z<a)or (z>b)`	**a.** `z<a || z>b`
b. `(z>a)or(z>b)`	**b.** `z>a || z>b`
c. `(z<a)and(z>b)`	**c.** `z<a && z>b`
d. `(z>=a)and(z<=b)`	**d.** `z>=a && z<=b`

52. Stabiliţi care dintre următoarele expresii este adevărată dacă şi numai dacă numerele reale strict pozitive **a,b,c** reprezintă lungimile laturilor unui triunghi dreptunghic:

a. `(c*c=a*a+b*b)or(a*a=b*b+c*c) or(b*b=a*a+c*c)`	**a.** `(c*c==a*a+b*b)||(a*a==b*b+ c*c)||(b*b==a*a+c*c)`
b. `(c*c=a*a+b*b)and(c>=b>=a)`	**b.** `(c*c==a*a+b*b)&&(c>=b>=a)`
c. `(c*c=a*a+b*b)and(a*a=b*b+ c*c)and(b*b=a*a+c*c)`	**c.** `(c*c==a*a+b*b)&&(a*a==b*b+ c*c)&&(b*b==a*a+c*c)`
d. `c*c=a*a+b*b`	**d.** `c*c==a*a+b*b`

53. Stabiliţi care dintre următoarele expresii este adevărată dacă şi numai dacă numărul întreg **x** este par şi strict pozitiv:

a. `(x mod 2=1)and(x>0)`	**a.** `!((x%2!=0)&&(x<0))`
b. `(x mod 2=0)or(x>0)`	**b.** `(x%2==1)&&(x>0)`
c. `not((x mod 2<>0)or(x<=0))`	**c.** `!((x%2!=0)||(x<=0))`
d. `not((x mod 2<>0)and(x<0))`	**d.** `(x%2==0)||(x>0)`

54. Care din următoarele expresii reprezintă ultima cifră a numărului natural **x**?

a. `x – x div (10*10)`	**a.** `x – x / (10*10)`
b. `x div 10`	**b.** `x / 10`
c. `x – (x div 10)* 10`	**c.** `x – (x / 10) * 10`
d. `x – x mod 10`	**d.** `x – x % 10`

55. Care dintre următoarele expresii reprezintă media aritmetică a patru numere reale memorate în variabilele **a,b,c** şi **d**?

a. `(a+b+c+d)*0.25`
b. `((a+b)/2+(c+d)/2)/4`
c. `a+b+c+d/4`
d. `(a+b+c+d)*0.4`

56. Dacă **n** este un număr natural de exact două cifre **n=**$\overline{ab}$, definim răsturnatul lui **n** ca fiind numărul $\overline{ba}$ dacă **b≠0** şi respectiv **a** dacă **b=0**. De exemplu, răsturnatul lui **12** este **21**, iar răsturnatul lui **10** este **1**. Care dintre următoarele expresii reprezintă răsturnatul unui număr natural **n** cu exact două cifre?

a. `10*n div 10+n mod 10`
b. `n div 10*10+n mod 10`
c. `10*n mod 10+n div 10`
d. `n mod 10*10+n div 10`

a. `10*n/10+n%10`
b. `n/10*10+n%10`
c. `10*n%10+n/10`
d. `n%10*10+n/10`

57. Dacă **n** este o variabilă de tip **integer/int** ce reţine un număr natural cu exact **3** cifre, atunci care dintre următoarele expresii reprezintă cifra zecilor lui **n**?

a. `n div 100 mod 10`
b. `n mod 100 div 10`
c. `n div 10`
d. `n mod 10*10`

a. `n/100%10`
b. `n%100/10`
c. `n/10`
d. `n%10*10`

58. Variabila întreagă **nr** trebuie să reţină câţi multipli mai mari decât **0** şi mai mici sau egali decât **n** are numărul **k** (**n** şi **k** sunt numere naturale date). Care este expresia cu care trebuie completată atribuirea **nr:=... / nr=...?**

a. `n mod k`
b. `n-n div k`
c. `n div k`
d. `n-n mod k`

a. `n%k`
b. `n-n/k`
c. `n/k`
d. `n-n%k`

59. Un program urmează să atribuie variabilei **s** de tip **int** suma a două variabile de tp **int**, **a** şi **b**, care se citesc de la tastatură. Citirea variabilelor **a** şi **b** se face corect cu instrucţiunea:

a. `readln(a,b)`
b. `readln(a+b)`
c. `readln(S=a+b)`
d. `writeln(a,b)`

a. `cin >>a>> b ;  / scanf("%d%d",&a, &b);`
b. `cin >>a+b ;  / scanf("%d",&(a+b));`
c. `cin>> S= a + b ;  / scanf("%d%d",S = a+b);`
d. `cout<<a<< b ;  / printf("a=%d b=%d",a,b);`

60. Stabiliţi care dintre următoarele expresii atribuie variabilei **m** de tip **float** valoarea mediei aritmetice a numerelor întregi **x** şi **y**:

a. `m:=x+y/2;`
b. `m:=(x+y)/2;`
c. `m:=x div 2+y div 2;`
d. `m=(x+y)/2;`

a. `m=x+y/2;`
b. `m=(x+y)/2.0;`
c. `m=x/2+y/2;`
d. `m==(x+y)/2;`

61. După care dintre următoarele secvenţe de instrucţiuni, ambele variabile întregi **x** şi **y** vor reţine valoarea **-1**, indiferent de valorile pe care le reţineau înaintea executării secvenţei?

a.
```
y :=x; x :=-1
```
b.
```
x :=1; y :=-x
```
c.
```
x :=-1; y :=-x
```
d.
```
x :=-1; y :=x
```

a.
```
y=x; x=-1 ;
```
b.
```
x=1; y=-x ;
```
c.
```
x=-1; y=-x ;
```
d.
```
x=-1; y=x ;
```

62. Care dintre următoarele variante inserează o cifră **c** în faţa ultimei cifre a unui număr natural **n**?

a.
```
n:=(n mod 10*10+c)*10+n div 10
```
b.
```
n:=n div 10+c+n mod 10
```
c.
```
n:=(n div 10*10+c)*10+n mod 10
```
d.
```
n:=(n div 10+c)*10+n mod 10
```

a.
```
n=(n%10*10+c)*10+n/10;
```
b.
```
n=n/10+c+n%10;
```
c.
```
n=(n/10*10+c)*10+n%10;
```
d.
```
n=(n/10+c)*10+n%10;
```

63. Variabila întreagă **a** reţine un număr natural format din exact două cifre. Care dintre următoarele instrucţiuni atribuie variabilei întregi **b** o valoare egală cu suma cifrelor numărului memorat în variabila **a**?

a.
```
b:=a mod 100
```
b.
```
b:=a mod 10+a div 10
```
c.
```
b:=a mod 10 + a div 100
```
d.
```
b:=a mod 2 + a div 2
```

a.
```
b = a%100;
```
b.
```
b = a%10 + a/10;
```
c.
```
b = a%10 + a/100;
```
d.
```
b = a%2 + a/2;
```

64. Care este valoarea expresiei următoare?

```
7 div 5*5
```
a. 0 **b.** 7

```
7/5*5
```
c. 5 **d.** 0.28

65. Fie expresia: $(x+y^2)^2+z(x+y)$. Care este varianta corectă ce reprezintă transcrierea acestei expresii în cadrul unui program **Pascal/C/C++**?

a.
```
sqrt(x+sqrt(y))+z*(x+y)
```
b.
```
sqr(x+sqrt(y))+z*(x+y)
```
c.
```
sqr(x+y*y)+z*(x+y)
```
d.
```
sqr(x+sqr(y))+z(x+y)
```

a.
```
pow(x+y*y,2)+z(x+y)
```
b.
```
sqrt(x+sqrt(y))+z*(x+y)
```
c.
```
pow(x+pow(y,2),2)+z*(x+y)
```
d.
```
pow(x+pow(y))+z*(x+y)
```

66. Cum se scrie în limbajul Pascal/C/C++ atribuirea din pseudocod alăturată? Variabilele **a**,**n**,**x** şi **y** sunt de tip real iar **x** şi **y** sunt nenule.

$$a \leftarrow \frac{2n}{x * y}$$

a.
```
a:= 2*n/x/y;
```
b.
```
a:= 2n/(x*y);
```
c.
```
a:= 2+n/x*y;
```
d.
```
a:= 2*n/x*y;
```

a.
```
a = 2*n/x/y;
```
b.
```
a = 2n/(x*y);
```
c.
```
a = 2+n/x*y;
```
d.
```
a = 2*n/x*y;
```

67. Care dintre următoarele expresii are ca valoare o treime din valoarea variabilei reale **a**?

a. `a/(3*2)/2` **b.** `a/3+a/2` **c.** `a/2/3+a/3/2` **d.** `a/(2/3)/3`

68. Ştiind că variabilele **a,b** şi **c** de tip întreg memorează valorile **a=2, b=30, c=3,** stabiliţi care este rezultatul evaluării expresiei aritmetice **a+b+c+b/a/c** ?

a. **95** b. **80** c. **38** d. **40**

69. Care dintre următoarele atribuiri elimină cifra din mijloc a unui număr natural **n** cu exact **5** cifre?

a. `n:=n mod 1000*100+n div 100`
b. `n:=n div 1000*100+n mod 100`
c. `n:=n div 1000+n mod 100`
d. `n:=n div 100*100+n mod 100`

a. `n=n%1000*100+n/100;`
b. `n=n/1000*100+n%100;`
c. `n=n/1000+n%100;`
d. `n=n/100*100+n%100;`

70. Ştiind că variabilele **a** şi **b** sunt utilizate pentru a memora două numere naturale cu cel puţin două cifre fiecare, stabiliţi care dintre instrucţiunile de mai jos determină, în urma executării, iniţializarea variabilei **m** cu cifra zecilor a numărului obţinut prin adunarea numerelor memorate în **a** şi **b.**

a. `m:=a mod 100 div 10+b mod 100 div 10`
b. `m:=(a+b) mod 100`
c. `m:=(a+b) div 10 mod 100`
d. `m:=(a+b) mod 100 div 10`

a. `m=a%100/10+b%100/10;`
b. `m=(a+b)%100;`
c. `m=(a+b)/10%100;`
d. `m=(a+b)%100/10;`

71. Ştiind că numărul natural **n** are valoarea **8473**, care din următoarele instrucţiuni atribuie variabilei întregi **a** valoarea **47**?

a. `a:=n mod 100`
b. `a:=(n mod 1000) div 10`
c. `a:=(n div 1000) mod 10`
d. `a:=((n div 10)*100) div 100`

a. `a=n%100`
b. `a=(n%1000)/10`
c. `a=(n/1000)%10`
d. `a=((n/10)*100)/100`

72. Fie **n** un număr natural cu cinci cifre. Care dintre variantele de mai jos determină, în urma executării, eliminarea din numărul **n** a cifrei sutelor?

a. `n:=n div 1000;`
b.
```
x:=n div 100;
n:=x div 10+n div 1000;
```
c. `a:=n mod 100; n:=n div 1000+a;`
d. `n:=n div 1000*100+n mod 100;`

a. `n=n/1000;`
b. `x=n/100; n=x/10+n/1000;`
c. `a=n%100; n=n/1000+a;`
d. `n=n/1000*100+n%100;`

73. Ştiind că variabilele **a** şi **b** sunt utilizate pentru a memora două numere reale, stabiliţi care dintre secvenţele de instrucţiuni de mai jos determină, în urma executării, iniţializarea variabilei **m** cu diferenţa absolută a valorilor memorate în **a** şi **b**.

a. `m:=a-b`	**a.** `m=a-b;`
b. `m:=a; if m<a then m:=b-m`	**b.** `m=a; if(m<a) m=b-m;`
c. `m:=a; if m>b then m:=a-m`	**c.** `m=a; if(m>b) m=a-m;`
d. `m:=a-b; if a<b then m:=-m`	**d.** `m=a-b; if(a<b) m=-m;`

74. Care au fost valorile variabilelor **x** şi **y**, de tip **integer/int**, la începutul executării secvenţei de instrucţiuni alăturată dacă la finalul executării **x** are valoarea **2007** iar **y** are valoarea **2009**?

`x:=x-1; y:=2*x+y; x:=2*x+1;`	`x=x-1; y=2*x+y; x=2*x+1;`

a. **x=1002** şi **y=5** **b.** **x=1004** şi **y=3**

c. **x=3** şi **y=1004** **d.** **x=2007** şi **y=2009**

75. Fie **n** un număr natural format din **6** cifre. Pentru a memora în variabila **a** cifra miilor numărului **n**, se foloseşte atribuirea :

a. `a:=n div 1000 mod 10`	**a.** `a=n/1000%10`
b. `a:=n div 100 div 10`	**b.** `a=n/100/10`
c. `a:=n mod 1000 mod 100`	**c.** `a=n%1000%100`
d. `a:= n div 1000 mod 100`	**d.** `a=n/1000%100`

76. Ştiind că variabila **a** este de tip **integer/int**, variabila **d** este de tip **boolean/int**, iar variabilele **b** şi **c** sunt de tip **real/float**, care dintre următoarele instrucţiuni de atribuire nu este corectă:

a. `a:=trunc(sqrt(a*a))`	**a.** `a=sqrt(a*a);`
b. `c:=2*b*a mod 2`	**b.** `c=2*b*a%2;`
c. `d:=b<c`	**c.** `d=(b<c);`
d. `b:=b-c`	**d.** `b=b-c;`

77. Care dintre următoarele variabile nu-şi modifică valoarea în urma executării secvenţei de instrucţiuni:

`c:=2*a+b; a:=a+b;` `a:=c-a; b:=c-2*a`	`c=2*a+b; a=a+b;` `a=c-a; b=c-2*a`

indiferent de valorile lor iniţiale? (**a**,**b**,**c** sunt variabile de tip întreg).

a. doar **c** **b.** doar **a** şi **b** **c.** toate **d.** niciuna

78. Care dintre următoarele variabile nu-şi modifică valoarea în urma executării secvenţei de instrucţiuni:

`c:=a-b; a:=b+c; b:=a-c;`	`c=a-b; a=b+c; b=a-c;`

indiferent de valorile lor iniţiale? (`a`,`b`,`c` sunt variabile de tip întreg).

a. `c` **b.** niciuna **c.** doar `a` şi `b` **d.** toate

79. Ştiind că:

`a` este de tip `integer`	`a` şi `d` sunt de tip `int`
`d` este de tip `boolean`	`b` şi `c` sunt de tip `float`
`b` şi `c` sunt de tip `real`	

care dintre următoarele instrucţiuni de atribuire **nu** este corectă:

a. `b:=b-2*c;`	**a.** `b=b-2*c;`
b. `d:=b>=c;`	**b.** `d=(b>=c);`
c. `c:=2+b mod 2*a;`	**c.** `c=2+b%2*a;`
d. `b:=sqrt(a*a);`	**d.** `b=sqrt(a*a);`

80. Ce se va afişa după executarea următoarei secvenţe de instrucţiuni?

`a:=3; b:=4; a:=a-b;`
`a:=a+b; a:=b-a;`
`write(a,' ',b);`
	`a=3; b=4; a=a-b; b=a+b; a=b-a;`
	`cout<<a<<" "<<b;`
	`/ printf("%d %d",a,b);`

a. `3 3` **b.** `4 4` **c.** `3 4` **d.** `4 3`

81. Care dintre următoarele secvenţe realizează interschimbarea valorilor variabilelor `a` şi `b` (numere întregi de cel mult `3` cifre)?

a. `a:=b; b:=a`	a. `a=b; b=a;`
b. `a:=aux; a:=b; b:=aux`	b. `a=aux; a=b; b=aux;`
c. `a:=a+b; b:=a-b; a:=a-b`	c. `a=a+b; b=a-b; a=a-b;`
d. `aux:=a; b:=aux;a:=b`	d. `aux=a; b=aux;a=b;`

82. Fie `n` un număr natural de cel puţin `4` cifre. Secvenţa care atribuie variabilei întregi `c` cifra miilor numărului natural `n` este:

a. `c:=n div 1000`	a. `c=n/1000;`
b. `c:=n div 10000 mod 1000`	b. `c=n/10000%1000;`
c. `c:=n mod 10000 div 1000`	c. `c=n%10000/1000;`
d. `c:=n mod 10000`	d. `c=n%10000;`

83. Care dintre următoarele secvenţe interschimbă corect valorile variabilelor `a` şi `b`; se ştie că `a, b` şi `aux` sunt variabile numerice de acelaşi tip?

a. `aux :=b; b :=a; a :=aux;`	**a.** `aux = b; b = a; a = aux;`
b. `aux :=a; b :=a; b :=aux;`	**b.** `aux = a; b = a; b = aux;`

c.	`aux :=b; aux :=a; a :=b;`	**c.**	`aux = b; aux = a; a = b;`
d.	`b :=aux; aux :=a; a :=b;`	**d.**	`b = aux; aux = a; a = b;`

84. Fie un număr **x** care aparţine inervalului **[101,120]**. Care este numărul minim de numere pe care trebuie să le testăm dacă sunt divizori ai lui **x** pentru a putea afirma cu siguranţă că **x** este număr prim?

a.	**4 numere**	**b.**	**[x/2]-1** numere (unde **[x/2]** este partea întreagă a valorii **x/2**)
c.	**9** numere	**d.**	**10** numere

85. Ce se afişează în urma executării secvenţei următoare de instrucţiuni dacă **x** este o variabilă reală, iar **n** este o variabilă întreagă?

```
x:=12.34;
n:=(trunc(x*10)) mod 10;
write(n);
```

```
x=12.34;
n=((int)(x*10))%10;
cout<<n; / printf("%d", n);
```

a. 4 **b.** 1 **c.** 2 **d.** 3

86. Care este valoarea tipărită de secvenţa următoare de program?

```
var a,b:integer;f:real;
begin a:=25;b:=7; f:=a/b;
 f:=trunc(f*100);
 f:=f/100; write(f:0:2) end.
```

```
int a=25,b=7;
float f; f=(float)a/b;
f=(int)(f*100); f=f/100;
cout<<f; /printf("%0.2f",f);
```

a. 3 **b.** 3.57 **c.** 3.5714 **d.** 35.71

87. Pentru care dintre următoarele valori ale variabilei **x** secvenţa de program alăturată afişează mesajul **NU**?

```
if x>0 then
  if x<3 then
      write('DA')
 else write('NU')
```

```
if (x>0)
 if(x<3)
   cout<<"DA"; / printf("DA");
 else cout<<"NU";/ printf("DA");
```

a. 3 **b.** 1 **c.** 2 **d.** 4

88. În ce situaţie, secvenţa următoare va afişa pe ecran **două** cifre?

```
If a>b Then  Writeln('1');
If a<b Then  Writeln('2')
Else  Writeln('0');
```

```
if(a>b)cout<<"1"; / printf("1");
if(a<b)cout<<"2"; / printf("2");
else cout<<"0"; / printf("0");
```

a. Dacă **a=b** **b.** **Dacă a>b** **c.** Dacă **a<b** **d.** Niciodată
Dacă **a==b**

89. Care dintre secvenţele de mai jos este echivalentă cu secvenţa următoare din punct de vedere al valorii pe care o primeşte variabila **g**? **a**, **x**, **y** şi **g** sunt variabile de acealaşi tip întreg.

```
if a>0 then
 if x<>y g:=1
  else g=2
else g=3
```

a.
```
if (a>0) and (x<>y) then  g=1
  else g=2
       else g=3
```

b.
```
if (a>0) and (x<>y) then g=1
  else if a>0 then g=3
        else g=2
```

c.
```
if (a>0) and (x<>y) then g=1
  else if a>0 then g=2
         else g=3
```

d.
```
if (a>0) or (x<>y) then g=1
  else if a>0 then g=3
         else g=2
```

```
if (a>0)
 if (x!=y) g=1;
  else g=2;
else g=3;
```

a.
```
if (a>0 && x!=y) g=1;
  else g=3;
       else g=2;
```

b.
```
if (a>0 && x!=y) g=1;
  else if (a>0) g=3;
       else g=2;
```

c.
```
if (a>0 && x!=y) g=1;
  else if (a>0) g=2;
       else g=3;
```

d.
```
if (a>0 || x!=y) g=1;
  else if (a>0) g=3;
       else g=2;
```

90. O secvenţă de instrucţiuni echivalentă cu secvenţa următoare care să conţină o singură instrucţiune **if** este:

```
if x>y then
  if y>z then
    if z>x then s:=x+y+z
           else p:=x*y*z
```

a.
```
if(x>y)and(y>z) then s:=x+y+z
                else p:=x*y*z
```

b.
```
if(x>y)and(y>z)then s:=x+y+z
```

c.
```
if(x>y)or(y>z)then s:=x+y+z
```

d.
```
if(x>y)and(y>z)then p:=x*y*z
```

```
if(x>y)
 if(y>z)
   if(z>x)s=x+y+z;
     else p=x*y*z;
```

a.
```
if(x>y&&y>z) s=x+y+z;
          else p=x*y*z;
```

b.
```
if(x>y&&y>z) s=x+y+z;
```

c.
```
if(x>y||y>z) s=x+y+z;
```

d.
```
if(x>y&&y>z) p=x*y*z;
```

91. Fie variabilele **x** şi **y** de tipul **word** ale căror valori se presupun cunoscute**.** Care dintre următoarele secvenţe afişează mesajul **DA** dacă ambele numere sunt pare ?

a.
```
if x*y mod 2=0 then write ('DA');
```

b.
```
if (x mod 2 <>0 ) or (y mod 2<>0)
then  write ('DA');
```

c.
```
if not((x mod 2<>0)or(y mod  <>0))
  then write ('DA');
```

d.
```
if (x mod 2 <>0 ) and (y mod 2<>0)
      then  write ('DA');
```

```
if(x*y%2==0)
  cout<<"Da";
     / printf("DA ") ;
if(x%2 !=0 || y%2 !=0)
   cout<< "DA " ;
     / printf("DA ") ;
if( !(x%2 !=0 || y%2 !=0))
   cout<<"DA " ;
    / printf("DA ") ;
if(x%2 && y%2)
   cout<<"DA";
    / printf("DA ") ;
```

92. Care este cea mai mică valoare pozitivă pe care o poate memora variabila întreagă **x** astfel încât în urma executării instrucţiunii următoare să se afişeze valoarea lui **x**.

```
if ( x > 71 mod 2+3 div x)
       then write(x);
```

```
if (x>71%2+3/x)
   printf("%d",x);
          / cout<<x;
```

a. 2 **b.** 3 **c.** 4 **d.** 5

93. Câte valori afişează următoarea secvenţă dacă pentru variabila întreagă **x** se citeşte valoarea **10072**?

```
readln(x);
while x>0 do begin
  writeln(x);
  x := x div 100
end
```

```
cin>>x; / scanf("%d",&x);
while (x>0)
{ printf("%d ",x);/ cout<<x<<" ";
 x=x/100;
}
```

a. 3 **b.** 1 **c.** 2 **d.** 5

94. Ce valori vor avea variabilele de tip întreg **x** şi **y** după executarea secvenţei următoare?

```
x:=1; y:=11;
while(x<=y) do
 begin x:=x+1; y:=y-1 end
```

```
x=1; y=11;
while(x<=y)
   { x=x+1; y=y-1;}
```

a. x=5 y=7 **b.** x=7 y=5 **c.** x=6 y=5 **d.** x=6 y=6

95. Pentru care din următoarele valori ale variabilei **n** secvenţa de program alăturată următoare valoarea **0** în urma executării ei:

```
while (n mod 10>=2) do
  n:=n div 10;
write(n);
```

```
while(n%10>=2)
  n=n/10;
printf("%d",n); /  cout<<n;
```

a. 1111 **b.** 9282 **c.** 3003 **d.** 1345

96. Ştiind că valoarea iniţială a variabilei **k** este un număr natural par cu cel mult **4** cifre, stabiliţi valoarea tipărită de secvenţa următoare.

```
c:=0;
while n MOD 10=0 do
begin n:=n DIV 10; c:=c+1
end;
```

```
c=0;
while (n%10==0)
   {n=n/10;c++;}
```

a. 1 **b.** 5 **c.** 0 **d.** 4

97. Care este valoarea iniţială a variabilei **n** astfel încât, la sfârşitul executării secvenţei următoare, variabila întreagă **c** să aibă valoarea **3**?

```
while k>1 do k:=k-2;
n:=abs(k-5); write(n)
```

```
while (k>1) k=k-2;
n=abs(k-5); cout<<n;
```

a. 123 **b.** 10020 **c.** 5000 **d.** 10001

98. Ce valoare are variabila întreagă **n** în urma executării secvenţei următoare?

```
n:=0;
while n<=11 do n:=n+2;
```

```
n=0 ;
while(n<=11) n=n+2;
```

a. 11 **b.** 9 **c.** 10 **d.** 12

99. Următoarea secvenţa de program, va afişa :

```
a:=99;
while (a>=1) do
  begin   write(a);   a:=a-2
end
```

```
a=99;
while (a>=1)
{cout<<a;
    / printf("%d",a);
 a=a-2; }
```

a. toate numerele naturale de două cifre.
b. numerele naturale impare mai mici decât **100**
c. toate numerele întregi mai mici decât **99**
d. numerele naturale pare, mai mari decât **1**

100. Se consideră secvenţa de instrucţiuni următoare în care variabilele **i** şi **j** sunt de tip întreg. Stabiliţi care dintre următoarele valori poate fi valoare iniţială pentru variabila **j** astfel încât executarea secvenţei să se realizeze în timp finit.

```
i:=0;
while (i+j<=10) do
begin
    i:=i+1; j:=j-2
end
```

```
i=0;
while (i+j<=10)
  { i=i+1; j=j-2;}
```

a. 17 b. 6 c. 5 d. 1

101. Ce se va afişa în urma executării secvenţei de instrucţiuni următoare?

```
i:=1;j:=1;
while (i<=7) do
   begin
     write(j); i:=i+3
end
```

```
i=1;j=1;
while(i<=7)
{ printf("%d",j);/cout<<j;
 i=i+3;
}
```

a. 1 1 1 **b.** 1 2 3 4 5 6 7
c. 1 4 7 **d.** 1 1 1 1 1 1 1

102. Care trebuie să fie valoarea iniţială a variabilei întregi **i** pentru ca următoarea secvenţă să afişeze şirul **XXX**?

```
while (i<>3) do
  begin
    i:=i-1; writeln('XX')
  end
```

```
while (i!=3)
{ i=i-1;
  printf("XX");/cout<<"XX";
}
```

a. 3 b. nu există nici o valoare
c. 1 d. 2

103. Care trebuie să fie valoarea iniţială a variabilei **i** de tip întreg pentru ca în urma executării instrucţiunii următoare, pe ecran să fie afişată secvenţa de caractere ***** ?

```
while (i*5<1000)  do
 begin
  write ('*'); i:=i*2+10
 end
```

```
while (i*5<1000)
{ printf(„*");/cout<<"*";
  i=i*2+10;
}
```

a. **3** b. **11** c. **13** d. **5**

104. Considerând secvenţa de program următoare şi ştiind că de la tastatură se citeşte valoarea **234**, ce valoare se afişează pe ecran după executarea secvenţei date?

```
read(n);  x:=1;
while n>0 do begin
  x:=x*n%10; n:=n div 10
end;
write(x)
```

```
cin>>n; / scanf("%d",&n);
x=1;
while (n>0)
    {x=x*n%10; n=n/10;}
cout<<x; / printf("%d",x);
```

a. **9** b. 4 c. **24** d. 0

105. Se consideră secvenţa următoare de instrucţiuni, unde **i** este o variabilă de tip întreg. Cu ce valoare trebuie completate punctele de suspensie astfel încât să se afişeze **8** caractere ***** (adică ********)?

```
i:=....;
while i<10 do begin
  write('**');
  i:=i+1
end;
```

```
i=....;
while (i<10)
{cout<<"**"; / printf("**");
 i++;
}
```

a. 4 **b. 5** **c. 6** **d. 7**

106. Câte atribuiri se execută în secvenţa următoare, pentru **n=245**?

```
s:=0;
while n<>0 do begin
   s:=s+1; n:=n div 100 end;
```

```
s=0;
while (n!=0)
    {s=s+1; n=n/100;}
```

a. 5 **b. 7** **c. 3** **d. 1**

107. În secvenţa de instrucţiuni următoare, **n** şi **y** sunt variabile întregi. Valoarea variabilei **y** la finalul executării secvenţei este:

```
n:=156; y:=770;
 while n*y>0 do
   if n>y then n:=n mod y
      else y:=y mod n;
 y:=y+n
```

```
n=156; y=770;
while(n*y>0)
  if (n>y) n=n%y;
     else y=y%n;
y=y+n;
```

a. 13 **b. 0** **c. 2** **d. 4**

108. Se consideră următoarele declarări:

```
var x,i:integer;
```

```
int x=3,i=0;
```

Ce va afişa secvenţa de mai jos?

```
x:=3;i:=0;
while x-1<>0 do
   begin x:=x-1;i:=i+1;end;
write(i);
```

```
while(x-1){x--;i++;}
cout<<i;
   / printf("%d",i);
```

a. 1 **b.** 0 **c.** 2 **d.** 4

109. De câte ori se execută instrucţiunea de afişare în următoarea secvenţă de instrucţiuni, unde **i** este o variabilă de tip întreg?

```
i:=3;
while i<=9 do
  i:=i+1;
write(i);
```

```
i=3;
while (i<=9)
  i++;
cout<<i; / printf ("%d", i);
```

a. 6 **b.** 3 **c.** 1 **d.** 7

110. Dacă **n** este un număr natural , ce realizează următoarea secvenţă?

```
p:=1;
while n<>0 do
 begin p:=p*n;n:=n-1; end;
```

```
p=1;
while(n) p=p*(n--);
```

a. calculează în variabila p valoarea n*p
b. calculează în variabila p valoarea n^p
c. calculează în variabila p valoarea lui n!
d. calculează în variabila p valoarea p^n

111. Care este valoarea expresiei **2+n** după executarea secvenţei de program următoare ?

```
n:=100;
while n>=2 do n:=n-1;
```

```
n=100;
while (n>=2) n--;
```

a. 3 **b.** 1 **c.** 4 **d.** 2

112. Care dintre următoarele afirmaţii, referitoare la secvenţa de instrucţiuni de mai jos, este adevarată?

```
if (a>10)then
begin b:=7; c:=8;end;
  while (a > b) do begin
    b:=3;c:=c+1; write(c);end;
```

```
if (a>10){b=7; c=8;}
while (a > b)
{ b+=3; c++;
  printf("%d",c); /cout<<c; }
```

a. Secvenţa conţine o structură de decizie care este inclusă într-o structură repetitivă.
b. Secvenţa conţine o structură repetitivă care este inclusă într-o structură de decizie.

c. Secvenţa conţine o structură de decizie, urmată de o structură repetitivă, urmată de o instrucţiune de afişare.
d. Secvenţa conţine o structură de decizie urmată de o structură repetitivă.

113. Care dintre următoarele secvenţe de instrucţiuni determină afişarea pe ecran , în urma executării, a numărului **55**(**i** şi **j** fiind variabile de tip **integer/int**)?

Varianta Pascal

a.
```
i:=5; j:=6; while(j>4)do write(j); j:=j-1;
```
b.
```
i:=5; j:=6; while(j>4)do begin write(i); j:=j-1;end;
```
c.
```
j:=5; for i:=5 to 5 do write(i);
```
d.
```
j:=5; for i:=1 to 1 do write(j);
```

Varianta C/C++

a.
```
i=5; j=6; while(j>4) printf("%d",j);/cout<<j; j--;
```
b.
```
i=5; j=6; while(j>4){printf("%d",i);/cout<<i; j--;}
```
c.
```
j=5; for(i=5;i<=5;i++) printf("%d",i);/cout<<i;
```
d.
```
j=5; for(i=1;i<2;i++) printf("%d",j);/cout<<j;
```

114. De câte ori se execută instrucţiunea de afişare în următoarea secvenţă de instrucţiuni, ştiind că **i** şi **j** sunt variabile de tip întreg?

```
for i:=3 to 8 do
  for j:=i+1 to 9 do
    writeln(i, ' ', j);
```

```
for (i=3; i<=8; i++)
  for (j=i+1; j<=9; j++)
   cout<<i<<" "<<j<<endl;
    / printf ("%d %d \n",i, j);
```

a. 15 **b.** 21 **c.** 6 **d.** 9

115. Care din următoarele instrucţiuni va afişa în ordine descrescătoare toate numerele naturale impare mai mici sau egale cu o valoare naturală dată **a**?

a.
```
for i:=1 downto a do
  if a mod 2=1 then
      write(i,' ')
```
b.
```
for i:=1 downto a do
  if a mod 2=0 then
      write(i,' ')
```
c.
```
for i:=a downto 1 do
  if a mod 2<>0 then
      write(i,' ')
```
d.
```
for i:=a to 1 do
  if a mod 2=1 then
      write(i,' ')
```

a.
```
for (i=1;i>=a;i=i+2)
 cout<<i<<" ";
   / printf("%d ",i);
```
b.
```
for (i=1;i<=a;i--)
 if (a%2==0) cout<<i<<" ";
   / printf("%d ",i);
```
c.
```
for (i=a;i>=1;i--)
 if (i%2!=0) cout<<i<<" ";
    / printf("%d ",i);
```
d.
```
for (i=a;i<=1;i++)
 if (a%2==1)cout<<i<<" ";
   / printf("%d ",i);
```

116. Care este valoarea variabilei **x** după executarea următoarei secvenţe de instrucţiuni?

```
x:=0;
for i:=1 to 10 do ;
x:=x+1;
```

```
x=0;
for(i=1;i<=10;i++);
x=x+1;
```

a. 0 **b.** eroare de sintaxă **c.** 10 **d.** 1

117. În secvenţa următoare **x % y** semnifică restul împărţirii întregi a lui **x** la **y**, iar **x/y** câtul împărţirii întregi a lui **x** la **y**.
Pentru **n>2**, natural, secvenţa următoare afişează **1** dacă şi numai dacă:

```
for i:=2 to n div 2 do
if n mod i=0 then ok:=0
   else ok:=1;
write(ok)
```

```
for (i=2 ; i<=n/2; i++)
if (n%i==0) ok=0;
   else ok=1;
printf("%d",ok); /cout<<ok;
```

a. numărul **n** nu este divizibil cu **n div 2**
b. numărul **n** nu este prim
c. numărul **n** este par
d. numărul **n** este prim

118. Ce valoare are variabila întreagă **n** în urma executării secvenţei de mai jos, ştiind că iniţial valoarea ei este **0**?

```
for i:=1 to 100 do
  for j:=1 to i do n:=n+1;
```

```
for(i=1;i<=100;i++)
  for(j=1;j<=i;j++) n=n+1;
```

a. 500 **b.** 150 **c.** 1000 **d.** 5050

119. Care dintre următoarele secvenţe de instrucţiuni afişează toate numerele naturale din intervalul **[1,20]** care **nu** sunt divizibile cu **3** ?

a.
```
for i:=1 to 20 do
write(i,' ');
```

b.
```
for i:=1 to 20 do
 if i mod 3=0 then
   write(i,' ');
```

c.
```
for i:=1 to 20 do
 if(i mod 3=1)or(i mod 3=2)
    then  write(i,' ');
```

d.
```
for i:=3 to 20 do
 write(i,' ');
```

a.
```
for(i=1;i<=20;i++)
 cout<<i<<' ';
       / printf("%d ",i);
```

b.
```
for(i=1;i<=20;i++)
  if(i%3==0)
    cout<<i<<' ';
    / printf("%d ",i);
```

c.
```
for(i=1;i<=20;i++)
 if(i%3==1|| i%3==2)
  cout<<i<<' ';
  / printf("%d ",i);
```

d.
```
for(i=3;i<=20; i++)
 cout<<i<<' ';
  / printf("%d ",i);
```

120. Secvenţa de program următoare va afişa :

```
var c:char;
for c:='A' to 'Z' do write(c);
```

```
char c;
for(c='A';c<='Z';c++)
  cout<<c;/
     printf("%c",c);
```

a. numerele naturale din intervalul **[1,27]**
b. numerele naturale din intervalul **[65,90]**
c. literele mari ale alfabetului englez
d. codurile **ASCII** ale literelor mari din alfabetul englez

121. Ce se va afişa în urma execuţiei secvenţei de cod următoare, pentru **n = 6**, dacă **n** şi **p** sunt variabile de tip întreg ?

```
p := 1;
for i := n downto 2 do
      p := p * i;
write(p);
```

```
p = 1;
for (i = n; i > 1; i--)
          p *= i;
printf("%d", p); | cout << p;
```

a. 20 **b.** 120 **c.** 720 **d.** 21

122. Se consideră secvenţa de program de mai jos. Instrucţiunea de afişare se execută de:

```
for i:=1 to 10 do
  for j:=i+1 to 10 do
    write(j);
```

```
for(i=1;i≤10;i++)
for(j=i+1;j≤10;j++)
   cout<<j;/
      printf(”%d”,j);
```

a. 100 ori **b.** 10 ori **c.** 20 ori **d.** 45 ori

123. Se consideră următoarele declarări:

```
const x:vector=array[0..4] of
              integer =(0,1,5,3,4);
var y,i:integer;
```

```
int  x[5]={0,1,5,3,4};
int y,i;
```

Ce va afişa secvenţa de mai jos?

```
y:=x[1];
for i:=0 to 4 do
   if y<x[i] then  y:=x[i];
write(y);
```

```
y=x[1];
for(i=0;i<=4;i++)
  if(y<x[i]) y=x[i];
cout<<y; / printf(”%d”,y);
```

a. 0 **b.** 5 **c.** 13 **d.** nedeterminată

124. Ce se va afişa pe ecran în urma executării următoarelor instrucţiuni, dacă pentru variabila întreagă **a** se citesc, în ordine, numerele: **1234, 234, 52, 25, 5432, 819** ?

```
for i:=1 to 6 do
 begin  read(a);
  if i mod 2=0 then
    write(a div 100 mod 10)
  else
    write(a div 10  mod 10)
 end
```

```
for(i=1;i<=6;i++)
{ scanf(“%d”,&a);/ cin>>a;
  if(i%2==0)
    printf(“%d”,a/100%10);
        / cout<<a/100%10;
  else  printf(“%d”,a/10%10);
        / cout<<a/10%10;
}
```

a. 230241 **b.** 432221 **c.** 220241 **d.** 325038

125. Ce se va afişa pe ecran în urma executării următoarelor instrucţiuni ?

```
for i:=1 to 5 do
 for j:=5 downto i do
      if i mod 2 = 0
        then write(i)
        else write(j)
```

```
for(i=1;i<=5;i++)
 for(j=5;j>=i;j--)
   if(i%2==0)
     printf("%d",i); / cout<<i;
   else  printf("%d",j); / cout<<j;
```

a. 12345
b. 111115432333545
c. 543212222543445
d. 122333444455555

126. De câte ori se va executa instrucţiunea de decizie din secvenţa de program următoare, dacă valoarea variabilei întregi **n** este 8?

```
for i:= 1 to n-1 do
 for j:= i to n-1 do
  if j mod i = 0
   then write(i,j);
```

```
for (i = 1; i < n; i++)
 for (j = i; j < n; j++)
  if (j % i == 0)
   printf("%d%d", i, j); | cout<<i<<j;
```

a. 16 **b.** 38 **c.** 28 **d.** 36

127. În secvenţa de instrucţiuni următoare, variabilele **n, x** şi **y** sunt de tip întreg. Dacă valoarea variabilei **n** este un număr natural nenul, de câte ori este evaluată expresia logică **x<=n** în timpul executării secvenţei?

```
x:=1;  y:=x-1;
repeat
   y:=x*(x-1)+y; x:=x+1
until x>n
```

```
x=1; y=x-1;
do  { y=x*(x-1)+y;
      x=x+1; }
while(x<=n);
```

a. de n^2 ori **b.** de **n** ori **c.** o singură dată **d.** de **n+1** ori

128. În secvenţa următoare variabilele **a, b** şi **s** sunt de tip **integer/int**. Ce valoare va memora variabila **a** după executarea secvenţei?

```
a:=25; b:=8; s:=0;
 repeat
   s:=s+1; a:=a-b
 until (a<b)
```

```
a=25; b=8; s=0;
do
 { s=s+1; a=a-b;}
while (a>=b);
```

a. 1 **b.** 24 **c.** 3 **d.** 0

129. Dacă variabilele **a** şi **b** sunt de tip **integer/int**, ce valori vor avea variabilele **a** şi **b** la finalul executării secvenţei de instrucţiuni următoare?

```
a:=5; b:=0;
repeat
  a:=a-1; b:=b+a*a
until a=0;
```

```
a=5; b=0;
do
   { a=a-1; b=b+a*a;}
while(a!=0);
```

a. a=0 şi b=0
b. a=-1 şi b=25
c. a=0 şi b=55
d. a=0 şi b=30

130. Ce valori vor avea variabilele întregi **a** şi **b** la finalul executării secvenţei de instrucţiuni următoare?

```
a:=1; b:=0;
repeat
    b:=b+1; a:=a*b
until a>=125
```

```
a=1; b=0;
do
  { b=b+1; a=a*b; }
while(a<125);
```

a. a=126 şi b=6
b. a=720 şi b=6
c. a=125 şi b=5
d. a=720 şi b=125

131. Care este cea mai mare valoare iniţială a variabilei întregi **ind**, pentru care următoarea secvenţă de program va afişa exact un caracter **'*'** ?

```
a := 5;
repeat
 write('*'); ind:=ind+1
until a > ind;
```

```
a = 5;
do {printf("*"); |cout << '*';
    ind++;}
while (a <= ind);
```

a. 5 **b.** 4 **c.** 6 **d.** 3

132. După executarea secvenţei de program următoare variabilele **a** şi **b** de tip **integer/int** vor avea valorile:

```
a:=1; b:=7;
repeat
      a:=a+1; b:=b-1
until(a>b)
```

```
a=1; b=7;
do{
      a++; b--;
}while(a<=b);
```

a. a=3 b=3 **b.** a=4 b=4 **c.** a=3 b=5 **d.** a=5 b=3

1.3. Probleme (programe pseudocod, Pascal | C / C++)

1. Se consideră programul pseudocod alăturat:

```
citeşte a, b
    (numere naturale)
┌dacă a<b atunci
│    t←a; a←b; b←t
└■
r←a%b
┌cât timp r≠0 execută
│   a←b; b←r; r←a%b
└■
scrie b
```

S-a notat cu **x%y** restul împărţirii lui **x** la **y**.

1) Ce se va afişa pentru **a=150**, **b=125**?
2) Ştiind că **a=15**, câte valori din intervalul închis **[10, 20]** pot fi introduse pentru variabila **b** astfel algoritmul să afişeze **1**?
3) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

2. Se consideră programul pseudocod alăturat:

```
citeşte a,b {a,b∈N}
┌dacă a<b atunci
│  a←a-b; b←a+b; a←b-a
└■
k←0
┌cât timp a≥b execută
│  a←a-b; k←k+2
└■
scrie k
```

1) Ce va afişa algoritmul pentru **a=3** şi **b=10**?
2) Scrieţi algoritmul pseudocod, echivalent cu algoritmul dat, care să folosească un alt tip de structură repetitivă.
3) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului.
4) Pentru câte perechi de valori **(a,b)**, cu **a** şi **b** aparţinând intervalului **[1,10]**, rezultatul afişat este egal cu **10**?

3. Se consideră programul pseudocod alăturat:

```
citeşte a,b {a,b∈N}
┌dacă a%2=0
│   atunci  a←a+1
└■
s←0
┌cât timp a≤b execută
│  a←a+2; s←s+1
└■
scrie s
```

S-a notat cu cu **x%y** restul împărţirii numerelor întregi **x** şi **y**.

1) Ce va tipări algoritmul pentru **a=2** şi **b=11**?
2) Scrieţi programul Pascal/C/C++ corespunzător algoritmului.
3) Scrieţi algoritmul pseudocod echivalent cu cel dat care foloseşte un alt tip de structură repetitivă.
4) Ştiind că **b** primeşte la citire valoarea **79**, determinaţi **2** valori distincte pe care le poate primi **a** şi pentru care rezultatul afişat este **40**.

4. Se consideră programul pseudocod alăturat:

```
citeşte n  {număr natural}
p←1; i←1
┌cât timp i<n şi p≠0 execută
│  i←i+1; x←p*i; p←x mod 10
└■
scrie p,i
```

S-a notat cu **x mod y** restul împărţirii numărului întreg **x** la numărul întreg **y**.

1) Ce se afişează pentru **n=6**?
2) Scrieţi o valoare pentru **n** astfel încât ambele valori afişate să fie nenule.
3) Scrieţi programul Pascal/C/C++ corespunzător algoritmul dat.
4) Scrieţi un program pseudocod echivalent cu algoritmul dat care să utilizeze un alt tip de structură repetitivă.

5. Se consideră programul pseudocod alăturat:

S-a notat cu **x mod y** restul împărţirii numărului întreg **x** la numărul întreg **y** şi cu **[z]** partea întreagă a numărului **z.**

1) Ce se afişează dacă se citesc valorile (în această ordine) **2576** şi **31465**?
2) Scrieţi două perechi de valori pentru **a** şi **b**, astfel încât să se afişeze mesajul **DA.**
3) Scrieţi programul Pascal/C/C++ corespunzător algoritmul dat.

```
citeşte a,b  {numere naturale}
x ←1
┌cât timp (a>0)şi(b>0) execută
│┌dacă (a mod 10)<(b mod 10)
│|  atunci x←0
│└■
| a←[a/10]; b←[b/10]
└■
┌dacă (x=1) şi (b=0)
|  atunci scrie "DA"
|  altfel scrie "NU"
└■
```

4) Scrieţi un algoritm pseudocod echivalent cu cel dat, care să utilizeze un alt tip de structură repetitivă**.**

6. Se consideră programul pseudocod alăturat:

S-a notat cu **x mod y** restul împărţirii numărului întreg **x** la numărul întreg **y** şi cu **[z]** partea întreagă a numărului **z.**

1) Ce se afisează pentru **n=35724**?
2) Scrieţi o valoare pentru **n** astfel încat să se afişeze mesajul **DA.**

```
citeşte n{număr natural}
b←n mod 10
┌ cât timp n>=10 execută
|  n← [n/10]
└■
┌ dacă b mod 2 = n mod 2
|   atunci scrie "DA"
|   altfel scrie "NU"
└■
```

3) Scrieţi programul Pascal/C/C++ conform algoritmului.
4) Scrieţi un program pseudocod echivalent cu algoritmul dat care să utilizeze un alt tip de structură repetitivă.

7. Se consideră programul pseudocod alăturat.S-a notat cu **x%y** restul împărţirii numerelor întregi **x** şi **y.**

1) Care sunt valorile afişate pentru următoarele date de intrare: **100,1,2,10,5,0 ?**
2) Scrieţi o secvenţă de valori pentru **x** astfel încât rezultatul afişat în urma executării algoritmului să fie **0 10.**

```
citeşte x {x natural}
nr←0
s←0
┌cât timp x≠0 executã
|   nr←nr+1
|   ┌dacã nr%2=0 atunci
|   |  s←s+x%10
|   └■
|   citeşte x
└■
scrie s,nr
```

3) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.
4) Scrieţi programul pseudocod care să fie echivalent cu algoritmul dat şi care să conţină o structură repetitivă cu test final.

8. Se consideră programul pseudocod alăturat:

S-a notat cu **x%y** restul împărţirii numărului întreg **x** la numărul întreg **y** şi cu **[z]** partea întreagă a numărului real **z.**

```
citeşte n {număr natural nenul}
i←1
┌cât timp n>0 execută
│ ┌dacă n%2>0 atunci scrie i
│ └■
│ i←i+1; n←[n/2]
└■
```

1) Ce se va afişa pentru **n=333**?

2) Scrieţi care este cea mai mică valoare strict pozitivă pentru variabila **n** astfel încât să se afişeze succesiunea de valori **12345**?

3) Scrieţi programul pseudocod care să fie echivalent cu algoritmul dat, dar în care să se înlocuiască structura **cât timp ... execută** cu o structura repetitivă cu test final.

4) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

9. Se consideră programul pseudocod alăturat.S-a notat cu **x%y** restul împărţirii numerelor întregi **x** şi **y** şi cu **[z]** partea întreagă a numărului real **z**.

```
citeşte x ,m
{x întreg, m natural}
y←1
┌ cât timp m>0 exec
│┌ dacă m%2 =0
│| atunci
│|   m←[m/2]; x←x*x
│| altfel
│|   m←m-1;y←y*x
│└■
└■
scrie y
```

1) Care este valoarea afişată pentru **x=-2** şi **m=9**?

2) Scrieţi o pereche de valori pentru **x** şi **m** astfel încât rezultatul afişat să fie **1.**

3) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

4) Scrieţi programul pseudocod care să fie echivalent cu algoritmul dat şi care să conţină o structură repetitivă cu test final.

10. Se consideră programul pseudocod alăturat în care s-a notat cu **x|y** relaţia "**x** divide pe **y**" sau "**y** este divizibil cu **x**".

```
citeşte n,k
(numere naturale nenule)
s←0
┌pentru i=1,n execută
│ ┌dacă k|i atunci s←s+i
│ └■
└■
scrie s
```

1) Ce se va afişa pentru **n=40,k=7**?

2) Determinaţi câte o valoare de două cifre pentru variabilele **n** şi **k** astfel încât rezultatul afişat să fie un număr impar.

3) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

4) Construiţi un algoritm echivalent fără a utiliza structuri repetitive, scriind programul Pascal/C/C++ corespunzător.

11. Se consideră programul pseudocod alăturat în care s-a folosit notaţia **[x]** pentru partea întreagă a lui **x**.

```
citeşte n (număr natural nenul)
t←1; c←n%10; n←[n/10]
┌cât timp t=1 şi n>0 execută
│ ┌dacă n%10>c atunci
│ │ t←0
│ └■
│ c←n%10; n←[n/10]
└■
scrie t
```

1) Ce se va afişa pentru **n=4357**?

2) Scrieţi o valoare de patru cifre pentru variabila **n** astfel încât rezultatul afişat să fie **1**.

3) Câte numere naturale nenule de cel mult două cifre pot fi introduse pentru variabila **n**, să se afişeze **1** pentru fiecare caz în parte.

4) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

12. Se consideră programul pseudocod alăturat în care s-a folosit notaţia **[x]** pentru partea întreagă a lui **x**.

```
citeşte n
(număr natural nenul)
p←1; s←0
┌cât timp p<=n execută
│ s←s+[n/p]; p←p*2
└■
scrie s
```

1) Ce se va afişa pentru **n=35**?

2) Scrieţi o valoare pentru variabila **n** astfel încât rezultatul afişat să fie **16**.

3) Scrieţi programul C sau C++ corespunzător algoritmului dat.

4) Scrieţi un număr natural nenul care nu poate fi afişat de program oricare ar fi valoarea naturală nenulă care se citeşte pentru variabila **n**.

13. Se consideră programul pseudocod alăturat:

```
citeşte n,m
    {numere naturale, m≤n}
s←0
┌cât timp n≥m execută
│  s←s+n
│  n←n-1
└■
scrie s
```

1) Ce se va afişa pentru **n=10** şi **m=4**?

2) Scrieţi o pereche de numere naturale de câte o cifră ce pot fi citite pentru variabilele **n** şi **m** astfel încât rezultatul afişat în urma executării algoritmului să fie **15**?

3) Scrieţi programul pseudocod care să fie echivalent cu algoritmul dat şi care să conţină o structură repetitivă cu test final.

4) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

14. Se consideră programul pseudocod alăturat:

S-a notat cu **x%y** restul împărţirii numărului întreg **x** la numărul întreg **y** şi cu **[z]** partea întreagă a numărului **z**.

1) Ce se va afişa pentru **n=2589**?
2) Scrieţi o valoare pentru variabila **n** astfel încât să se afişeze valoarea **0**.
3) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.
4) Scrieţi programul pseudocod care să fie echivalent cu algoritmul dat, dar în care să se înlocuiască structura **cât timp…execută** cu un alt tip de structura repetitivă.

```
citeşte n
      {număr natural}
ok←1
┌dacă n%10>[n/10]%10
│  atunci x←1
│  altfel x←0
└■
n←[n/10]
┌cât timp n>9 execută
│┌dacă n%10>[n/10]%10
│└   atunci y←1
│    altfel y←0
│└■
│┌dacă x ≠ y
│└   atunci ok←0
│└■
│ n←[n/10]
└■
scrie ok
```

15. Se consideră programul pseudocod alăturat:

S-a notat cu **x|y** relaţia „**x** divide pe **y**" sau „**y** este divizibil cu **x**" şi cu **[x]** partea întreagă a numărului real **x**.

1) Ce se va afişa pentru **n=112**?
2) Scrieţi o valoare pentru variabila **n** astfel încât să se afişeze o valoare egală cu **n**.
3) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.

```
citeşte n
i←2; p←1
┌cât timp n>1 execută
│ k←0
│ ┌cât timp i|n execută
│ │   k←i; n←[n/i]
│ └■
│ ┌dacă k≠0 atunci
│ │ p←p*k
│ └■
│ i←i+1
└■
scrie p
```

16. Se consideră programul pseudocod alăturat:

S-a notat cu **x%y** restul împărţirii lui **x** la **y** şi cu **[x]** partea întreagă a numărului real **x**

1) Ce se va afişa pentru **n=3533** şi **c=3**?

```
citeşte n,c
  (numere naturale,0≤c≤9 )
k←0
┌cât timp n%10=c execută
│  n←[n/10]; k←k+1
└■
scrie k
```

2) Scrieţi o valoare pentru **n** şi una pentru **c** astfel încât să se afişeze valoarea **0**.
3) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.
4) Scrieţi un algoritm echivalent cu algoritmul dat, dar care să utilizeze alt tip de structură repetitivă**.**

17. Se consideră programul pseudocod alăturat:

S-a notat cu **x%y** restul împărţirii lui **x** la **y**

```
citeşte a
(număr întreg)
k←0
┌cât timp a≠0 execută
│ citeşte b
│┌dacă a%2=b%2 atunci
│└ k←k+1
└■
 a←b
scrie k
```

1) Ce se va afişa dacă de la tastatură se introduce şirul de valori **2 4 6 5 7 3 9 8 0**?

2) Daţi un exemplu de şir de date de intrare cu cel puţin **3** elemente care să determine afişarea valorii **0**.

3) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.

4) Scrieţi un algoritm echivalent cu algoritmul dat, dar care utilizezează un alt tip de structură repetitivă.

18. Se consideră programul pseudocod alăturat:

S-a notat cu **[x]** partea întreagă a numărului real **x**.

```
citeşte a
   (a număr natural, a>1)
b←1/a; c←0
┌cât timp b<1 execută
│    b←b*10; c←c+1
└■
b←[b]
scrie c,b
```

1) Ce se va afişa pentru **a=12**?

2) Scrieţi o valoare pentru **a** astfel încât să se afişeze valorile **3** şi **1**.

3) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.

4) Scrieţi un algoritm echivalent cu algoritmul dat, dar care să utilizeze alt tip de structură repetitivă.

19. Se consideră programul pseudocod alăturat:

S-a notat cu **x%y** restul împărţirii lui **x** la **y** şi cu **[x]** partea întreagă a numărului real **x**.

```
citeşte n (număr natural cu
cel mult 9 cifre)
┌cât timp n≥10
│ s←0
│┌cât timp n≠0 execută
│└ s←s+n%10
│  n←[n/10]
└■
 n←s
scrie n
```

1) Ce se va afişa dacă valoarea citită pentru **n** este **989736** ?

2) Stabiliţi două numere diferite, de **5** cifre fiecare care, (atribuite iniţial lui **n**), au ca efect afişarea valorii **1**.

3) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.

4) Scrieţi un algoritm echivalent cu algoritmul dat, dar care să utilizeze alt tip de structură repetitivă.

20. Se consideră programul pseudocod alăturat:

S-a notat cu **x%y** restul împărţirii numărului natural **x** la numărul natural **y**, iar cu **[x]** partea întreagă a numărului real **x.**

```
citeşte n
  (număr natural, nenul)
p ← 1
┌cât timp p<n execută
│ n ← ([n/p]+1)*p+n%p
│ p ← p*10
└■
scrie n
```

1) Ce afişează algoritmul dacă se citeşte valoarea **15793**?

2) Care este valoarea care trebuie citită pentru ca în urma executării algoritmului dat să se afişeze **210**?

3) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.

4) Scrieţi un program pseudocod echivalent cu algoritmul dat în care structura **cât timp...execută** să fie înlocuită cu o structură repetitivă cu test final.

21. Se consideră programul pseudocod alăturat:

```
citeşte a,b,c (nr. reale,
a<>0)
┌cât timp c<>0 execută
│ d←b*b-4*a*c
│  ┌dacă d>0 atunci
│  │ scrie a,b,c
│  └■
│  a←b
│  b←c
│  citeşte c
└■
```

1) Ce se va afişa dacă **a=1, b=2, c=1** şi sunt citite valorile **2,-1,3,-2,5,0**?

2) Daţi un exemplu de trei valori reale pentru variabilele **a, b** şi **c** astfel încât structura **cât timp** să efectueze o singură iteraţie şi să afişeze trei valori pozitive.

3) Scrieţi un program pseudocod echivalent cu algoritmul dat care să conţină o structură repetitivă cu test final.

4) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.

22. Se consideră programul pseudocod alăturat:

```
citeşte n,m
       (nr. naturale nenule)
┌cât timp n<>m execută
│  ┌dacă n>m atunci n ← n-m
│  │          altfel m ← m-n
│  └■
└■
scrie n
```

1) Ce rezultat se afişează pe ecran pentru **n=12** şi **m=3** ?

2) Se citeşte pentru **n** valoarea **72**. Să se determine cea mai mică valoare de **3** cifre citită pentru **m** astfel încât să se afişeze valoarea **36**.

3) Scrieţi un program pseudocod echivalent cu algoritmul dat care să conţină o structură repetitivă cu test final.

4) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.

23. **Se consideră programul pseudocod alăturat,** unde subprogramul **suma(n)** returnează suma cifrelor numărului natural n transmis ca parametru.

```
citeşte n (n∈N)
nr←0
┌cât timp n>9 execută
│   n←suma(n)
│   nr←nr+1
└■
scrie nr
```

1) Care este valoarea afişată pentru **n=1999**?

2) Daţi exemplu de o valoare pentru **n** astfel încât valoarea afişată să fie **1**.

3) Care este cea mai mare valoare de patru cifre ce trebuie citită pentru variabila **n** astfel încât să se afişeze **3**?

4) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

24. **Se consideră programul pseudocod alăturat în care** s-a notat cu **x%y** restul împărţirii numărului natural **x** la numărul natural **y** şi cu **[z]** partea întreagă a numărului real **z**.

```
citeşte n
    {n număr natural}
z ←0
┌cât timp n>0 execută
│  c ←n%10; n←[n/10]
│   ┌dacă c%2=0
│   │ atunci z←z*10+c
│   └■
└■
scrie z
```

1) Care este valoare afişată pentru **n=52381**?

2) Scrieţi o valoare pentru variabila n astfel încât să se afişeze valoarea **0**.

3) Scrieţi un program pseudocod echivalent cu algoritmul dat care să conţină o structură repetitivă cu test final.

4) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.

25. **Se consideră programul pseudocod alăturat în care** s-a notat cu **x%y** restul împărţirii întregi a lui **x** la **y** şi cu **[x]** partea întreagă a numărului real **x**.

```
citeşte n  (nr. natural,n>1)
d←2 (d număr natural)
┌ cat timp n%d≠0 execută
│ d←d+1
└■
┌ cat timp n%d=0 execută
│ n←[n/d]
└■
┌ dacă n=1 atunci scrie d
│          altfel scrie n
└■
```

1) Ce se va afişa pentru **n = 45**?

2) Scrieţi o valoare cu două cifre care poate fi introdusă pentru variabila **n** astfel încât să se afişeze valoarea **2**.

3) Câte valori distincte poate primi variabila **n** astfel încât să se afişeze valoarea **4**?

4) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

26. Se consideră programul pseudocod alăturat:

```
citeşte a (număr natural)
b←0
┌cât timp a>0 execută
│┌dacă a%2>0 atunci
│└  b←b*10+a%10
└a←[a/10]
scrie b
```

S-a notat cu **x%y** restul împărţirii numerelor întregi **x** şi **y** şi cu **[x]** partea întreagă a numărului real **x**.

1) Care este valoarea afişată pentru **a=1789**?
2) Determinaţi cea mai mare valoare întreagă, formata din patru cifre pentru variabila **a** astfel încât rezultatul afişat să fie **15.**
3) Câte valori distincte, numere naturale, cuprinse intre **0** şi **50**, inclusiv, poate să primească variabila **a** pentru ca algoritmul sa afişeze valoarea **0**?
4) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.

27. Se consideră programul pseudocod alăturat:

```
citeşte n (număr natural)
i←0; j←1
┌cât timp j≤n execută
│ k←i; i←j; j←i+k
└
scrie j
```

1) Care este valoarea afişată pentru **n=20**?
2) Determinaţi cea mai mică valoare naturală a variabilei **n** astfel încât rezultatul afişat să fie **34.**
3) Pentru câte valori naturale distincte ale variabilei **n**, algoritmul afişează **13**.
4) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.

28. Se consideră programul pseudocod alăturat:

```
citeşte n (nr. natural)
s1 ← 0; s2 ← 0
nr ← 0
┌cât timp n<>0 execută
│ ┌dacă n % 2 = 0 atunci
│ │    s1 ← s1 + n % 10
│ │ altfel s2←s2+n % 10
│ └
│ n ← [n/10]
└
┌dacă s1 = s2 atunci
│  nr ← 1
└
scrie nr
```

S-a notat cu **x%y** restul împărţirii numerelor întregi **x** şi **y** şi cu **[x]** partea întreagă a numărului real **x**.

1) Care este valoarea afişată pentru **n=50324**?
2) Pentru **n** $=\overline{31a2b}$, unde **a** este cifra sutelor iar **b** este cifra unităţilor, câte perechi ordonate (**a,b**) de cifre există pentru ca valoarea afişată să fie **1**.
3) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.
4) Să se scrie un program pseudocod echivalent cu cel dat folosindu-se un alt tip de structură repetitivă.

29. Se consideră programul pseudocod alăturat:

```
citeşte n {nr. natural}
max ← 0
┌cât timp n≠0 execută
│n ← [n/10]
│┌dacă max<n%10 atunci
│|  max ←n%10
│└■
└■
scrie max
```

S-a notat cu **x%y** restul împărţirii numerelor întregi **x** şi **y** şi cu **[x]** partea întreagă a numărului real **x**.

1) Ce se afişează dacă se citeşte **17358**?

2) Scrieţi câte numere naturale de trei cifre pot fi introduse pentru variabila **n** astfel încât rezultatul afişat să fie **2**?

3) Scrieţi programul pseudocod care să fie echivalent cu algoritmul dat şi care să conţină o structură repetitivă cu test final.

4) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.

30. Se consideră programul pseudocod alăturat în care s-a folosit notaţia **x%y** pentru restul împărţirii întregi a lui **x** la **y** şi cu **[a]** partea întreagă a numărului real **a**.

```
citeşte n (număr natural nenul)
s1←0
s2←0
┌cat timp n>0 executa
|  s1←s1+n%10
|  n←[n/10]
|  s2←s2+n%10
|  n←[n/10]
└■
┌dacă s1=s2 atunci
|       scrie 1
|altfel
|       scrie 0
└■
```

1) Ce se va afişa pentru **n=12345**?

2) Scrieţi o valoare cu două cifre care poate fi introdusă pentru variabila **n** astfel încât să se afişeze valoarea **1**.

3) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

4) Câte valori distincte cu două cifre pot fi introduse pentru variabila **n** astfel încât să se afişeze valoarea **1**?

31. Se consideră programul pseudocod alăturat :

```
citeşte x,y
      (numere naturale , x<y)
k←0
┌cat timp x<y execută
|x←x+1; y←y-1; k←k+1
└■
┌dacă x=y atunci
|       scrie 2*k+1
|altfel
|       scrie 2*k
└■
```

1) Ce se va afişa pentru **x=1**,**y=10**?

2) Câte perechi **(x,y)** există în intervalul **[1;10]** astfel încât să se afişeze valoarea **5**?

3) Scrieţi programul Pascal/C corespunzător algoritmului dat.

4) Scrieţi programul pseudocod care să fie echivalent cu algoritmul dat şi care să nu conţină nici o structură repetitivă sau recursivă.

32. Se consideră programul pseudocod alăturat:

S-a notat cu **x%y** restul împărţirii numărului natural **x** la numărul natural nenul **y**.

1) Ce se afişează dacă **x = 2** şi **y = 9**?
2) Ce valoare trebuie introdusă pentru variabila **x** dacă valoarea citită pentru **y** este **4** şi algoritmul afişează **256**?
3) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.
4) Scrieţi un algoritm pseudocod echivalent cu cel dat în care să nu se utilizeze nicio structură **dacă**.

```
citeşte x, y
p ← 1
┌cât timp y > 0 execută
│ ┌dacă y % 2 = 0 atunci
│ │    y ← y – 2
│ │    p ← p * x * x
│ │ altfel
│ │    y ← y – 1
│ │    p ← p * x
│ └■
└■
scrie p
```

33. Se consideră programul pseudocod alăturat:

S-a notat cu **x%y** restul împărţirii numărului natural **x** la numărul natural nenul **y**.

1) Ce se afişează pentru **n=15**?
2) Determinaţi câte valori de două cifre se pot introduce pentru variabila **n** astfel încât să se afişeze numai valoarea **1**.

```
citeşte n (număr natural)
i←1
┌cât timp i≤n execută
│┌dacă n%i=0
│|  atunci scrie i;i←i+1
│|  altfel i←i+2
│└■
└■
```

3) Dorim să înlocuim structura de decizie din algoritmul dat cu secvenţa următoare astfel încât algoritmul obţinut să fie echivalent cu cel dat. Cu ce instrucţiune putem să completăm punctele de suspensie?

```
┌dacă n%i=0 atunci
│  scrie i
│  ...
└■
 i← i+2
```

4) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.

34. Se consideră programul pseudocod alăturat:

S-a notat cu **x%y** restul împărţirii lui **x** la **y** şi cu **[x]** partea întreagă a numărului real **x**.

1) Care este valoarea afişată pentru **x=783851**?
2) Daţi un exemplu de valoare care, dacă este citită atunci algoritmul afişează valorile **5 3**.
3) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.

```
m←0; k←0
citeşte x (x nr natural)
┌cât timp x>0 execută
│ c←x%10; x←[x/10]
│ ┌dacă c>m atunci
│ │ m←c; k←1
│ │altfel
│ │   ┌dacă c=m atunci
│ │   │ k←k+1
│ │   └■
│ └■
└■
scrie m,k
```

35. Se consideră programul pseudocod alăturat:

S-a notat cu **a<-->b** operaţia de interschimbare a valorilor variabilelor **a** şi **b** şi cu **[x]** partea întreagă a numărului real memorat în variabila **x**.

1) Ce se afişează dacă se citesc valorile **56398** şi **398560** pentru **a** şi respectiv **b**?
2) Dacă **a=5100**, determinaţi o valoare de 3 cifre pentru variabila **b** astfel încât să se afişeze **DA**.
3) Scrieţi programul **Pascal/C/C++** corespunzător.
4) Scrieţi programul pseudocod sau **Pascal/C/C++** care să fie echivalent cu programul dat, cu excepţia cazurilor în care valorile iniţiale ale variabilelor **a** şi **b** sunt egale. În aceste cazuri, trebuie să se afişeze **NU**.

```
citeşte a,b (numere naturale)
dacă a<b atunci
          a<-->b
┌cât timp a>b execută
│  a←[a/10]
└■
┌dacă a=b atunci
│  scrie 'DA'
│altfel
│  scrie 'NU'
└■
```

36. Se consideră programul pseudocod alăturat în care s-a folosit notaţia **[a]** pentru partea întreagă a numărului real **a**.

1) Care este valoarea afişată pentru **n=1234**?
2) Scrieţi o valoare de două cifre pentru variabila **n** astfel încât să se afişeze **1**.
3) Pentru câte valori distincte ale lui **n**, număr natural cu maximum **3** cifre se afişează valoarea **0**?
4) Scrieţi programul Pascal/C corespunzător algoritmului dat.

```
citeşte n (număr întreg, n>0)
k←0; c←0
naux←n
┌cat timp naux>0 execută
│naux←[naux/10]
│k←k+1
│c←c*10+1
└■
┌pentru i←1,k execută
│n←n-c; c←[c/10]
└■
scrie n
```

37. Se consideră programul pseudocod alăturat:

S-a notat cu **x%y** restul împărţirii numerelor întregi **x** şi **y** şi cu **[x]** partea întreagă a numărului real **x**.

1) Ce va tipări algoritmul pentru **2793**?
2) Scrieţi programul **PASCAL/C/C++** corespunzător algoritmului.
3) Scrieţi algoritmul pseudocod, echivalent cu cel dat, care foloseşte un alt tip de structură repetitivă.
4) Daţi exemplu de o valoare nenulă pentru **n**, astfel încât rezultatul afişat să fie **0**.

```
citeşte n {n ∈ N}
a←n%10; m←a
┌cât timp n>9 execută
│ n←[n/10]; b←n%10
│ ┌dacă a>b atunci
│ │     m←m*10+b; a←b
│ └■
└■
scrie m
```

38. Se consideră programul pseudocod alăturat:

S-a notat cu **x%y** restul împărţirii numărului natural **x** la numărul natural **y**, iar cu **[x]** partea întreagă a numărului real **x.**

```
citeşte a
(a număr natural, a>1)
n←0
d←2
┌cât timp d≤a execută
│ i←0
│ ┌cât timp a%d=0 atunci
│ │ a←[a/d]; i←1
│ └■
│ n←n+i; d←d+1
└■
scrie n
```

1) Ce se afişează dacă numărul citit este **6**?
2) Care este cel mai mic număr care trebuie citit astfel încât să se afişeze valoarea **3**?
3) Scrieţi programul PASCAL/C/C++ corespunzător algoritmului dat.
4) Scrieţi un program pseudocod echivalent cu cel dat în care fiecare structură **cât timp...execută** să fie înlocuită cu o structură repetitivă cu test final.

39. Se consideră programul pseudocod alăturat:

```
┌citeşte a
│   (număr natural,a>0)
│ b←(a+2)*(a+3); k←0
│┌cât timp (b-a≥0)execută
│ b←b-a; k←k+1
│└■
│ scrie b,k
└■
```

1) Care sunt valorile afişate pentru **a=7**?
2) Stabiliţi o valoare pentru variabila **a** astfel încât după executarea secvenţei, **b** să fie **0**.
3) Scrieţi programul **PASCAL/C/C++** corespunzător algoritmului dat.
4) Scrieţi un program **PASCAL/C/C++** echivalent cu algoritmul dat, dar în care să nu se utilizeze structuri repetitive.

40. Se consideră programul pseudocod alăturat:

S-a notat cu **x%y** restul împărţirii numerelor întregi **x** şi **y** şi cu **[x]** partea întreagă a numărului real **x**.

```
citeşte n (nr. natural)
i ←1
┌cât timp i<=n execută
│ citeşte x (nr. natural)
│  nr←0
│ ┌cât timp x>0 execută
│ │   nr ←nr*10+x%10
│ │   x ← [x/1000]
│ └■
│  scrie nr
│  i ←i+1
└■
```

1) Ce valori vor fi afişate pentru următoarele numere citite de la tastatură: **5, 12345, 2007, 31005,124, 9356**.
2) Scrieţi un set de date de intrare, distincte, astfel încât să se afişeze de trei ori numărul **71**.
3) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.
4) Scrieţi programul pseudocod care să fie echivalent cu algoritmul dat şi care să conţină o structură repetitivă cu număr cunoscut de paşi în loc de o structură repetitivă cu test iniţial.

41. Se consideră programul pseudocod alăturat:

```
citeşte a,b {a,b∈N}
k←1
┌cât timp a+b>0 execută
│ ┌dacă a mod 10≠b mod 10
│ │  atunci k←0
│ └■
│  a←a div 10
│  b←b div 10
└■
scrie k
```

S-a notat cu **a mod b** restul împărţirii lui **a** la **b** şi cu **a div b** câtul împărţirii lui **a** la **b**.

1) Ce se va afişa pentru **a=123** şi **b=213**?
2) Indicaţi o valoare pentru variabila **a** şi o valoare pentru variabila **b** astfel încât algoritmul să afişeze valoarea **1**.
3) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.
4) Scrieţi un program Pascal/C/C++ echivalent cu algoritmul dat care să nu folosească nici o instrucţiune repetitivă.

42. Se consideră programul pseudocod alăturat:

```
citeşte n {n∈N,n≥2}
k←2; t←n
┌cât timp t≠1 execută
│ ┌cât timp t mod k=0 execută
│ │    m←k; t←t div k
│ └■
│ k←k+1
└■
scrie m
```

S-a notat cu **a mod b** restul împărţirii lui **a** la **b** şi cu **a div b** câtul împărţirii lui **a** la **b**.

1) Ce se va afişa pentru **n=4840**?
2) Care este cea mai mică valoare formată din exact **3** cifre ce poate fi introdusă pentru variabila **n** astfel încât algoritmul să afişeze valoarea **7**?
3) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.
4) Înlocuiţi **doar** instrucţiunea **scrie m** din algoritmul dat cu una sau mai multe instrucţiuni astfel încât noul algoritm să afişeze mesajul **"DA"** în cazul în care numărul **n** este prim, respectiv mesajul **"NU"** în caz contrar.

43. Se consideră programul pseudocod alăturat:

```
citeşte n,k
(n,k numere naturale)
s←0
┌cât timp n>0 şi k>0 execută
│   c←n%10
│   ┌dacă c%2=0 atunci s←s+c
│   └■
│   n←[n/10];   k←k-1
└■
scrie s
```

S-a notat cu **x%y** restul împărţirii lui **x** la **y** şi cu **[x]** partea întreagă a numărului real **x**.

1) Care este valoarea afişată pentru **n=83425** şi **k=3**?
2) Pentru **k=3** stabiliţi o valoare nenulă pentru **n** astfel încât rezultatul afişat să fie **0**.

3) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.

4) Scrieţi un program **Pascal/C/C++** pentru un algoritm echivalent cu algoritmul dat, în care să se utilizeze structura repetitivă cu număr cunoscut de paşi (cu contor).

44. Se consideră programul pseudocod alăturat.

S-au folosit următoarele notaţii: **mod** pentru restul împărţirii întregi şi **div** pentru câtul împărţirii întregi.

1) Care este valoarea afişată de acest algoritm dacă se citesc următoarele valori: **2,15,78,3,0**

2) Daţi exemplu de un şir de **5** valori astfel încât să se afişeze valoarea **0**.

3) Rescrieţi programul pseudocod dat, modificând un număr minim de linii astfel încât programul obţinut să afişeze numărul de valori impare citite.

4) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

```
citeşte a {număr natural}
m←0
┌cât timp a<>0 execută
│    d←0
│ ┌cât timp a mod 2=0 atunci
│ │     d←d+1; a←a div 2
│ └■
│ ┌dacă d>m atunci m←d
│ └■
│  citeşte a {număr natural }
└■
scrie m
```

45. Se consideră programul pseudocod alăturat:

S-a notat cu **x%y** restul împărţirii numerelor întregi **x** şi **y** şi cu **[x]** partea întreagă a numărului real **x**.

1) Ce valoare se va afişa pentru **n=480**?

2) Scrieţi programul corespunzător algoritmului dat.

3) Stabiliţi câte valori din intervalul **[10,40]** pot fi introduse pentru variabila **n**, astfel încât după executarea programului, valoarea afişată să fie **1**.

```
citeşte n
   (număr natural)
s←0; f←2
┌cât timp n>1 execută
│ p←0
│┌cât timp n%f=0
│└┘          execută
│ n←[n/f]; p←p+1
│┌dacă p≠0 atunci s←s+p
│└■
│f←f+1
└■
scrie s
```

4) Daţi exemplu de trei valori diferite pentru n astfel încât rezultatul afişat de fiecare dată, să fie un număr mai mare decât **5**, acelaşi pentru toate cele trei valori stabilite pentru **n**.

46. Se consideră programul pseudocod alăturat:

```
citeşte n {n nr natural}
a←0; p←1
┌cât timp n≠0 execută
│ ┌cât timp n>9 execută
│ │ n←[n/10]
│ └■
│ a←n*p+a;  p←p*10
│ citeşte n
└■
scrie a
```

S-a notat cu **x%y** restul împărţirii lui **x** la **y** şi cu **[x]** partea întreagă a numărului real **x**.

1) Care este valoarea afişată dacă se introduc de la tastatură valorile **59**, **480**, **16**, **329**, **0**?
2) Daţi exemplu de un şir de valori ce trebuie citite astfel încât valoarea afişată să fie **123**?
3) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.
4) Scrieţi un algoritm echivalent cu algoritmul dat, care să folosească doar structuri repetitive cu test final.

47. Se consideră programul pseudocod alăturat:

```
citeşte a,b
(a,b nr. naturale, a<=b)
k←0
┌pentru i←a,b,1 execută
│ n←i; c←0
│ ┌cât timp n>0 execută
│ │┌dacă n%10%2=1 atunci
│ ││ c←1
│ │└■
│ │n←[n/10]
│ └■
│ ┌dacă c=0 atunci k←k+1
│ └■
└■
scrie k
```

S-a notat cu **x%y** restul împărţirii lui **x** la **y** şi cu **[x]** partea întreagă a numărului real **x**.

1) Ce se va afişa pentru **a=204** şi **b=212**?
2) Pentru **a=24** care sunt valorile care citite pentru **b** determină afişarea valorii **4**.
3) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.
4) Daţi un exemplu de valori pentru **a** şi **b** (**a<b**) astfel încât rezultatul afişat să fie **0**.

48. Se consideră programul pseudocod alăturat:

```
citeşte n (nr. natural)
nr ← 0; m ← 0
┌pentru i←1,n execută
│  citeşte x (nr. natural)
│  cif←x%10
│ ┌dacă cif>m atunci
│ │   m ← cif;nr ← 1
│ │ altfel
│ │ ┌dacă cif=m atunci nr←nr+1
│ │ └■
│ └■
└■
scrie nr
```

S-a notat cu **x%y** restul împărţirii numerelor întregi **x** şi **y**.

1) Ce valoare va fi afişată dacă se citesc următoarele valori: **5, 372, 477, 21, 27, 517**?
2) Pentru **n=5**, scrieţi cele 5 valori care se introduc, pe rând, pentru variabila **x** astfel încât rezultatul afişat să fie **4**.
3) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.
4) Scrieţi un program pseudocod echivalent cu algoritmul dat care să conţină un alt tip de structură repetitivă în loc de structura repetitivă cu număr cunoscut de paşi.

49. Se consideră programul pseudocod alăturat:

S-a notat cu **x%y** restul împărţirii numerelor întregi **x** şi **y** şi cu **[x]** partea întreagă a numărului real **x**.

1) Care este valoarea afişată pentru **a=30** şi **b=42**?
2) Ştiind că **b=39**, determinaţi cea mai mare valoare de maximum **2** cifre a variabilei **a** astfel încât rezultatul afişat să fie **1.**

```
citeşte a,b (numere naturale)
c←0
┌pentru i=1,a execută
│┌dacă b%i=0 atunci
│└┌dacă a%i=0 atunci c←i
│ └■
└■
┌dacă c>0 atunci scrie c
└■
```

3) Scrieţi un algoritm pseudocod, echivalent cu cel dat, care să utilizeze o structură repetitivă cu test iniţial în locul structurii repetitive folosite în acest algoritm.
4) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.

50. Se consideră programul pseudocod alăturat:

S-a notat cu **x%y** restul împărţirii numerelor întregi **x** şi **y.**

1) Care este valoarea afişată pentru **n=20** şi **m=30**?
2) Ştiind că **m=22,** determinaţi cea mai mică valoare întreagă şi pozitivă pentru variabila **n** astfel încât rezultatul afişat să fie **2.**

```
citeşte n,m (număr natural)
b←0
┌pentru x=n,m execută
│┌dacă x>=2 atunci
│└ a←2
└┌cât timp x%a>0 execută
 └   a←a+1
┌dacă x=a atunci b←b+1
└■
scrie b
```

3) Ştiind că **n=10**, precizaţi care este numărul valorilor distincte ale lui **m** astfel încât algoritmul să afişeze valoarea **2**.
4) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.

51. Se consideră programul pseudocod alăturat în care s-a notat cu **x|y** relaţia "**x** divide pe **y**" sau "**y** este divizibil cu **x**".

1) Ce se va afişa pentru **a=20,b=50,n=7**?
2) Pentru **n=5**, determinaţi câte o valoare de două cifre pentru fiecare dintre variabilele **a** şi **b** astfel încât rezultatul afişat să fie zero.

```
citeşte n,a,b
(întregi, a<b,n>0)
s←0
┌pentru i=a,b execută
│ ┌dacă n|i atunci
│ │ s←s+1
│ └■
└■
scrie s
```

3) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.
4) Scrieţi un program Pascal/C/ C++ care să fie echivalent cu algoritmul dat şi care să nu conţină nicio structură repetitivă.

52. Se consideră programul pseudocod alăturat:

S-a notat cu **x%y** restul împărţirii numărului natural **x** la numărul natural **y**, iar cu **[x]** partea întreagă a numărului real **x.**

```
citeşte n (număr natural)
pentru i←1,n execută
    p←1
    pentru j←i,2,-1 execută
        p←p*j
    scrie [p/(i*2)]
```

1) Ce se afişează dacă valoarea citită este 3?

2) Care este cea mai mică valoare citită pentru **n** astfel încât în şirul valorilor afişate să existe cel puţin 3 numere care au ultima cifră 0 şi care să se afle pe poziţii consecutive?

3) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.

4) Scrieţi un program pseudocod echivalent cu cel dat care să utilizeze o singură structură repetitivă.

53. Se consideră programul pseudocod alăturat:

S-a notat cu **[x]** partea întreagă a numărului real **x**.

```
citeşte n (n>6 natural)
S←0
pentru i←1,n execută
    S←S+i
    dacă S>10 atunci
        S←[S/2]
scrie S
```

1) Care este valoarea afişată dacă **n=10**?

2) Determinaţi o valoare pentru variabila **n** astfel încât să se afişeze valoarea 33.

3) Scrieţi un program pseudocod echivalent cu programul dat, care să nu conţină nici o structură repetitivă.

4) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

54. Se consideră programul pseudocod alăturat:

S-a notat cu **x%y** restul împărţirii numerelor întregi **x** şi **y** şi cu **[x]** partea întreagă a numărului real **x**.

```
citeşte n {n∈N*}
S←0
pentru i=1,n-1 execută
    pentru j=i+1,n execută
        S←S+1
scrie S
```

1) Ce va afişa algoritmul pentru **n=7**?

2) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului.

3) Scrieţi algoritmul pseudocod, echivalent cu cel dat care foloseşte numai structuri repetitive **cât timp**.

4) Scrieţi un algoritm echivalent cu cel dat, în limbaj pseudocod sau limbaj de programare, care să nu utilizeze nici o structură repetitivă.

55. Se consideră programul pseudocod alăturat:

S-a notat cu **x % y** restul împărţirii numărului natural **x** la numărul natural nenul **y**

1) Ce se va afişa dacă se citesc pentru **n** valoarea **5** şi pentru **x** valorile: **16, 8, 48, 0, 24**?
2) Dacă **n = 4**, daţi exemplu de patru valori pentru **x**, pentru care algoritmul scrie **1**.
3) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.
4) Scrieţi un algoritm pseudocod echivalent cu algoritmul dat care să utilizeze în locul structurii repetitive **repetă până când** o structură repetitivă condiţionată anterior.

```
citeşte n(număr natural)
d ← 0
pentru i ← 1, n execută
    citeşte x
  dacă d = 0 atunci
    d ← x
  altfel
  repetă
    r ← x % d; x ← d
    d ← r
  până când r = 0
   d ← x
scrie d
```

56. Se consideră programul pseudocod alăturat:

S-a notat cu **x%y** restul împărţirii numărului natural **x** la numărul natural nenul **y,** iar cu **[x/y]** câtul împărţirii

1) Ce se va afişa dacă se citesc pentru **n** valoarea **5** şi pentru **x,** în ordine, valorile: **123**, **22**, **5324**, **1**, **86935**?
2) Dacă **n = 4**, daţi valori pentru **x** astfel încât să se afişeze **1234**.
3) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.
4) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat, folosind în loc de structura **cât timp** alt tip de structură repetitivă.

```
s ← 0
citeşte n (număr natural)
pentru i ←1,n execută
 citeşte x(număr natural)
 nr ← x % 10
 cât timp x>9 execută
   nr ← nr * 10
   x ← [x/10]
 s ← s + nr
scrie s
```

57. Se consideră programul pseudocod alăturat:

S-a notat cu **a mod b** restul împărţirii lui **a** la **b** şi cu **a div b** câtul împărţirii lui **a** la **b.**

1) Ce se va afişa pentru **n=27155**?
2) Scrieţi o valoare formată din exact **4** cifre pentru variabila **n** astfel încât algoritmul să afişeze valoarea **4.**
3) Scrieţi un program pseudocod echivalent cu cel dat pentru toate valorile posibile ale lui **n** şi care să conţină o singură structură repetitivă.
4) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

```
citeşte n {n nr. natural}
k←0
pentru i←0,9 execută
  x←n
 cât timp x>0 execută
   dacă x mod 10=i atunci
     k←i
   x←x div 10
scrie k
```

58. **Se consideră programul pseudocod alăturat** în care s-a notat cu **n:3** faptul că **n** este divizibil cu **3**

1) Ce se va afişa dacă se citesc valorile **8** şi **17**?

2) Scrieţi toate perechile de valori de o cifră care dacă sunt citite în acest algoritm, determină fiecare afişarea valorii **0**.

3) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

```
citeşte a,b
   (numere naturale nenule, a≤b)
S ← 0
┌pentru n=a,b execută
│ ┌dacă n:3 atunci
│ │    S ← S - n
│ │ altfel
│ │    S ← S + n
│ └■
└■
scrie S
```

4) Scrieţi un program pseudocod care să fie echivalent cu algoritmul dat şi care să nu folosească structuri repetitive.

59. **Se consideră programul pseudocod alăturat:**

S-a notat cu **a mod b** restul împărţirii lui **a** la **b** şi cu **a div b** câtul împărţirii lui **a** la **b**.

1) Ce se va afişa pentru **n=6**?

```
citeşte n {n∈N}
┌pentru i←1,n-2 execută
│┌pentru j←i+1,n-1 execută
│ ┌pentru k←j+1,n executa
│ │┌dacă k*k=i*i+j*j
│ ││  atunci  scrie i,j,k
│ │└■
│ └■
└■
```

2) Care este cea mai mare valoare ce poate fi introdusă pentru **n** astfel încât algoritmul să afişeze doar tripletele **3,4,5** şi **6,8,10** ?

3) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

4) Scrieţi un program Pascal/C/C++ echivalent cu algoritmul dat care să folosească doar două structuri repetitive.

60. **Se consideră programul pseudocod alăturat:**

S-a notat cu **x%y** restul împărţirii numerelor întregi **x** şi **y** şi cu **[x]** partea întreagă a numărului real **x**.

1) Ce va tipări algoritmul pentru **a=132** şi **b=2464**?

2) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului.

```
citeşte a,b {a,b ∈ N}
s←0; x←a%10*10+a%10
y←b%10*10+b%10
┌pentru i=x,y execută
│┌dacă[i/10]=i%10
│ │   atunci s←s+1
│ └■
└■
scrie s
```

3) Scrieţi algoritmul pseudocod echivalent cu cel dat care foloseşte un alt tip de structură repetitivă.

4) Scrieţi un algoritm echivalent cu cel dat, în limbaj pseudocod sau limbaj de programare, care să nu utilizeze nici o structură repetitivă.

61. Se consideră programul pseudocod alăturat:

```
citeşte n
    {număr natural nenul}
s←0; x←2
pentru i←1,n execută
    s←s+x
    x←x+2
scrie s
```

1) Ce se va afişa pentru **n=8**?

2) Scrieţi o valoare strict pozitivă pentru variabila **n** astfel încât să se afişeze valoarea **132**?

3) Scrieţi programul pseudocod care să fie echivalent cu algoritmul dat şi care să nu conţină nicio structură repetitivă.

4) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat

62. Se consideră programul pseudocod alăturat:

S-a notat cu **x%y** restul împărţirii numărului natural **x** la numărul natural **y**.

```
citeşte a, b
    (numere naturale 0<a≤b)
pentru i←1,a execută
    dacă (a%i=0)şi(b%i=0)
      atunci
          x ← i
scrie x
```

1) Ce se va afişa pentru **a=10** şi **b=15**?

2) Scrieţi programul Pascal/C/C++ corespunzător cu algoritmul dat.

3) Explicaţi în 10-15 cuvinte ce reprezintă valoarea afişată de algoritm în raport cu valorile date.

4) Daţi un exemplu de valori pentru **a** şi **b** astfel încât algoritmul să afişeze **1**.

63. Se consideră programul pseudocod alăturat:

S-a notat cu **x%y** restul împărţirii numărului întreg **x** la numărul întreg **y.**

```
citeşte a,b
          {numere naturale}
c←a%10
pentru i←1,b-1 execută
  c←c*a; c←c%10
scrie c
```

1) Ce valoare afişează algoritmul pentru **a=28, b=10**?

2) Scrieţi o pereche de valori de câte două cifre pentru **a** şi **b** astfel încât algoritmul să afişeze valoarea **8**.

3) Scrieţi programul Pascal/C/C++ conform algoritmului dat.

4) Scrieţi programul pseudocod care să fie echivalent cu algoritmul dat şi care să conţină un alt tip de structură repetitivă.

64. Se consideră programul pseudocod alăturat:

```
citeşte a,n
                (numere naturale)
j←3
┌pentru i=1,n execută
│┌dacă i%2=0
│|    atunci a←a-j
│|    altfel a←a+j
│└■
│ j←7-j
└■
scrie a
```

S-a notat cu **x%y** restul împărţirii numărului natural **x** la numărul natural nenul **y**

1) Care este valoarea afişată dacă **a=12** şi **n=10**?
2) Daţi exemplu de număr natural care trebuie citit în variabila **n** astfel încât pentru **a=32,** să se afişeze valoarea **34**.
3) Scrieţi un program pseudocod echivalent cu programul dat, care să nu conţină nici o structură repetitivă.
4) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

65. Se consideră programul pseudocod alăturat:

```
citeşte a,n
                (numere naturale)
┌pentru i=1,n execută
│┌dacă i%2=0
│|  atunci  a←a-i
│|  altfel  a←a+i
│└■
└■
scrie a
```

S-a notat cu **x%y** restul împărţirii numărului natural **x** la numărul natural nenul **y**

1) Care este valoarea afişată dacă **a=12** şi **n=10**?
2) Determinaţi o valoare pentru variabila **n** astfel încât, pentru valoarea iniţială **a=32,** să se afişeze **34**.
3) Scrieţi un program pseudocod echivalent cu programul dat, care să nu conţină nici o structură repetitivă.
4) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat

66. Se consideră programul pseudocod alăturat, în care s-a notat cu **[x]** partea întreagă a numărului real **x**.

```
citeşte n (număr natural)
nr←0
┌pentru i=1,[n/2]execută
│ j←i; s←0
│ ┌cât timp s<n execută
│ │     s←s+j;  j←j+1
│ └■
│ ┌dacă s=n atunci nr←nr+1
│ └■
└■
scrie nr
```

1) Care este valoarea afişată pentru **n=30**?
2) Daţi exemplu de o valoare pentru **n** astfel încât valoarea afişată să fie **2**.
3) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.
4) Modificaţi structura **dacă...atunci** a programului astfel încât să se afişeze şi toate modurile în care poate fi descompus **n** ca sumă de numere naturale consecutive. Scrieţi programul pseudocod modificat astfel.

67. Se consideră programul pseudocod alăturat:

S-a notat cu **x%y** restul împărţirii numerelor întregi **x** şi **y** şi cu **[x]** partea întreagă a numărului real **x**.

1) Ce se va afişa pentru **a=10** şi **b=20**?

2) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

3) Scrieţi un algoritm echivalent cu algoritmul dat, în care să se utilizeze doar structuri repetitive cu test final.

```
citeşte a>b
    (numere naturale, a≤b)
k←0
pentru i←a,b execută
   nr←0;aux←i
   cât timp aux≠0 execută
    nr←nr*10+aux%10
    aux←[aux/10]
   ■
   dacă nr<i atunci
    k←k+1
   ■
■
scrie k
```

4) Daţi un enunţ de problemă care poate fi rezolvată cu acest algoritm.

68. Se consideră programul pseudocod alăturat în care s-a notat cu **x%y** restul împărţirii întregi a lui **x** la **y** şi cu **[x]** partea întreagă a numărului real **x**.

1) Ce se afişează pentru **n=23751**?

```
citeşte n (număr natural,n>0)
repeta
   b←n%10; n←[n/10]
pana cand b>=n%10
scrie n
```

2) Scrieţi o valoare cu trei cifre care poate fi introdusă pentru variabila **n** astfel încât să se afişeze valoarea **0**.

3) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

4) Adăugaţi o structură alternativă la sfârşitul algoritmului care să afişeze, în plus, mesajul **DA** dacă cifrele numărului **n** sunt în ordine strict descrescătoare şi mesajul **NU** în caz contrar.

69. Se consideră programul pseudocod alăturat: S-a notat cu **a%b** restul împărţirii numărului întreg **a** la numărul întreg **b**.

1) Scrieţi un şir de valori pentru variabila **x** astfel încât programul să afişeze, în urma executării, mesajul **'DA'**.

2) Scrieţi un şir de valori pentru variabila **x** astfel încât programul să afişeze, în urma executării, mesajul **'NU'**.

3) Scrieţi un program pseudocod echivalent cu algoritmul dat care să conţină o structură repetitivă cu test iniţial.

```
n←0
repetă
citeşte x {x număr natural}
   dacă x<>0 atunci
    dacă x%2=0
       atunci n←n+1
       altfel n←n-1
    ■
   ■
■ până când x=0
dacă n=0 atunci scrie 'DA'
         altfel scrie 'NU'
■
```

4) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.

70. Se consideră programul pseudocod alăturat:

S-a notat cu **x%y** restul împărţirii numărului întreg **x** la numărul întreg **y** şi cu **[z]** partea întreagă a numărului real **z.**

```
citeşte n  {număr natural }
x←0
┌repetă
│ x←x*10+n%10
│ n←[n/10]
│ e←n*(n-x)*([n/10]-x)
└până când e=0
```

1) Ce se va afişa pentru **n=123321**?

2) Scrieţi două valori, numere naturale consecutive formate fiecare din câte cinci cifre, pentru variabila **n** astfel încât rezultatele afişate să fie diferite.

```
┌dacă n>0
│ atunci scrie 'A'
│ altfel scrie 'B'
└■
```

3) Scrieţi programul pseudocod care să fie echivalent cu algoritmul dat şi care să conţină o structură repetitivă cu test iniţial.

4) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

71. Se consideră programul pseudocod alăturat:

S-a notat cu **x%y** restul împărţirii numărului natural **x** la numărul natural **y** şi cu **[z]** partea întreagă a numărului real **z.**

```
citeşte n  {număr natural }
x←0
p←1
┌repetă
│    x←x+(9-n%10)*p
│    n←[n/10]
│    p←p*10
└până când n=0

scrie x
```

1) Ce se va afişa pentru **n=102206**?

2) Scrieţi o valoare care poate fi introdusă pentru variabila **n** astfel încât rezultatul afişat în urma executării algoritmului să fie **7210**.

3) Scrieţi programul pseudocod care să fie echivalent cu algoritmul dat şi care să conţină o structură repetitivă cu test iniţial.

4) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

72. Se consideră programul pseudocod alăturat:

S-a notat cu **x%y** restul împărţirii numărului natural **x** la numărul natural **y**, iar cu **[x]** partea întreagă a numărului **x.**

```
citeşte a,b
(numere naturale nenule)
c←0
┌repetă
│  i←a%2
│  j←b%2
│ ┌dacă i+j=0 atunci
│ │ c←c+1
│ └■
│  a←a*i+(1-i)*[a/2]
│  b←b*j+(1-j)*[b/2]
└■ până când i*j=1
scrie c
```

1) Ce se afişează dacă numerele citite sunt **24** şi **36**?

2) Scrieţi două valori care trebuie citite (una pentru variabila **a** şi una pentru variabila **b**) astfel încât în urma executării algoritmului să se afişeze valoarea **0**.

3) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.

4) Scrieţi un program pseudocod echivalent cu cel dat în care structura **repetă...până când...** să fie înlocuită cu o structură repetitivă cu test iniţial.

73. Se consideră programul pseudocod alăturat în care s-a folosit notaţia **[x]** pentru partea întreagă a lui **x.**

1) Ce se va afişa pentru **n=875**?

```
citeşte n (număr natural)
┌repetă
│  n←[n/10]
└până când n<10
scrie n
```

2) Scrieţi algoritmul pseudocod echivalent cu algoritmul dat, dar care să utilizeze un alt tip de structură repetitivă.

3) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

4) Determinaţi câte numere naturale de cel mult două cifre pot fi introduse pentru variabila **n**, astfel încât rezultatul afişat să fie **3**.

74. Se consideră programul pseudocod alăturat. S-a notat cu **x%y** restul împărţirii numerelor întregi **x** şi **y** şi cu **[z]** partea întreagă a numărului real **z**

1) Ce valori se vor afişa pentru **n=40?**

2) Scrieţi o valoare a lui **n** pentru care, în urma executării algoritmului, singura valoare afişată este **n.**

```
citeşte n{n natural}
i←2

┌repetă
│┌cât timp n%i=0 execută
│||      scrie i
│||      n←[n/i]
│└■
│ i←i+1
└până când n=1
```

3) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

4) Scrieţi programul pseudocod care să fie echivalent cu algoritmul dat şi care să conţină o singură structură repetitivă.

75. Se consideră programul pseudocod alăturat:

S-a notat cu **x%y** restul împărţirii numerelor întregi **x** şi **y** şi cu **[x]** partea întreagă a numărului real **x**.

1) Care este valoarea afişată pentru **a=1775**?

2) Scrieţi cea mai mare valoare întreagă pentru variabila **a** astfel încât rezultatul afişat să fie **9.**

3) Precizaţi o valoare de patru cifre a variabilei **a** pentru care se execută doar o iteraţie a structurii **repetă-până când**.

```
citeşte a (număr natural)
a←a+1
┌repetă
│  a←a-1
│  b←a
│  s←0
│  ┌cât timp b≠0 execută
│  │  s←s*10+b%10
│  │  b←[b/10]
│  └■
└până când s=a
scrie a
```

4) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.

76. Se consideră programul pseudocod alăturat:

S-a notat cu **x%y** restul împărţirii numerelor întregi **x** şi **y** şi cu **[x]** partea întreagă a numărului real **x**.

```
citeşte n (nr natural)
x←0
┌repetă
│ x←x*10+n%10
│ n←[n/10]
└până când n=0
┌repetă
│ n←n*10+x%10
│ x←[x/100]
└până când x=0
scrie n
```

1) Ce se afişează dacă se citeşte de la tastatură numărul **n=29357**?

2) Scrieţi o valoare de **5** cifre pentru **n** astfel încât rezultatul afişat să fie **123**.

3) Scrieţi un program pseudocod echivalent cu algoritmul dat care să conţină alt tip de structuri repetitive.

4) Scrieţi un program **Pascal/C/C++** corespunzător algoritmului dat.

77. Se consideră programul pseudocod alăturat:

S-a notat cu **[x]** partea întreagă a numărului real **x**, şi cu **x%y** restul împărţirii lui **x** la **y**

```
citeşte a,b
(numere naturale)
 s←0
┌repetă
│ ┌dacă a%2≠0 atunci
│ │  s←s+b
│ └■
│ a←[a/2]; b←b*2
└■ până când a<1
scrie s
```

1) Care este valoarea afişată pentru **a=19** şi **b=45**?

2) Stabiliţi trei seturi de valori pentru variabilele **a** şi **b** astfel încât valoarea afişată pentru **s** să fie **480**.

3) Scrieţi programul corespunzător algoritmului dat.

4) Identificaţi o formulă de calcul pentru **s**. Scrieţi programul echivalent cu algoritmul dat folosind această formulă.

78. Se consideră programul pseudocod alăturat, unde **x div y** înseamnă câtul împărţirii numerelor întregi **x** şi **y**.

```
citeşte n (n∈N, n>9)
p←10; q←1
┌repetă
│   p←p*10; q←q*10
└până când q<=n şi n<=p
scrie n div q
```

1) Care este valoarea afişată pentru **n=81832** ?

2) Scrieţi o valoare pentru variabila **n** astfel încât să se afişeze valoarea **5**.

3) Câte valori distincte de trei cifre există pentru variabila **n** astfel încât să se afişeze **3**?

4) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

79. **Se consideră programul pseudocod alăturat**. S-au folosit urmatoarele notaţii: **x mod y** şi **x div y** pentru restul, respectiv câtul împărţirii întregi a lui **x** la **y** şi **|x|** pentru valoarea absolută a numărului **x**.

1) Ce se va afişa pentru **x=123476**?

2) Câte valori distincte de două cifre se pot introduce pentru variabila x astfel încât, de fiecare dată, să se afişeze mesajul „Da”?

3) Scrieţi programul pseudocod care să fie echivalent cu algoritmul dat, dar în care să se înlocuiască structura repetitivă **pentru** cu o structura repetitivă cu test final.

4) Scrieti programul Pascal/C/C++ **c**orespunzător algoritmului dat**.**

```
citeşte x{număr natural >1}
t←0;sp←0;si←0;
y←x;
┌repetă
│  t←t+1
│  x←x div 10
└până când x≠0
┌pentru p←t,1,-1 execută
│  c←y mod 10
│  y←y div 10
│ ┌dacă p mod 2=0
│ │  atunci sp←sp+c
│ │  altfel si←si+c
│ └■
└■
┌dacă |si-sp| mod 11=0
│   atunci scrie „DA”
│   altfel scrie „Nu”
└■
```

80. **Se consideră programul pseudocod alăturat.**

S-au folosit următoarele notaţii: **mod** pentru restul împărţirii întregi iar **div** pentru câtul împărţirii întregi.

1) Ce se va afişa pentru **x=1232189**?

2) Daţi o valoare pentru numărul **x** astfel încât algoritmul să nu afişeze nimic.

```
citeşte x {x nr.natural}
┌pentru c←0,9 execută
│  y←x; p←0
│ ┌repetă
│ │┌dacă y mod 10=c atunci
│ ││ p←p+1
│ │└■
│ │ y←y div 10
│ └până când y=0
│ ┌dacă p>1 atunciscrie c
│ └■
└■
```

3) Scrieţi un program pseudocod care să fie echivalent cu cel dat, dar în care să se înlocuiască structura repetitivă **pentru** cu o structură repetitivă cu test iniţial.

4) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

81. Se consideră programul pseudocod alăturat.

S-au folosit următoarele notaţii: **mod** pentru restul împărţirii întregi şi **div** pentru câtul împărţirii întregi.

```
citeşte x {nr.natural}
 v←0; z←1;
┌repetă
│ a←x mod 10
│ ┌dacă a>5  atunci v←v+z*5
│ │ altfel v←v+z*a
│ └■
│ x←x div 10;z←z*10
└până când x=0
 scrie v
```

1) Ce se va afişa pentru **x=1939**?

2) Indicaţi cea mai mare valoare posibilă pentru **x** astfel încât algoritmul să afişeze valoarea **2355**.

3) Înlocuind structura **dacă** cu secvenţa

```
┌dacă a>5  atunci
│...
└■
 v←...
```

trebuie să obţinem un algoritm echivalent cu cel dat. Cu ce trebuie înlocuite punctele de suspensie?

4) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

82. Se consideră programul pseudocod alăturat în care:

- **[x]** reprezintă partea întreagă a lui **x**
- **n:d** reprezintă faptul că **n** e divizibil cu **d**

```
citeşte n (număr natural, 1<n)
p←1
d←2
┌repetă
│ k←0
│ ┌cât timp n:d execută
│ │ n ← [n/d]; k ← k+1
│ └■
│ p ← p*(k+1); d ← d+1
└până când n=1
scrie p
```

1) Ce se va afişa pentru **n=20**?

2) Scrieţi o valoare de două cifre pentru variabila **n** astfel încât rezultatul afişat să fie impar.

3) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

4) Precizaţi câte valori există în intervalul închis **[20,40]** care, dacă sunt introduse pentru variabila **n**, rezultatul afişat este, de fiecare dată, egal cu **2**.

83. Se consideră programul pseudocod alăturat:

S-a notat cu **a mod b** restul împărţirii lui **a** la **b** şi cu **a div b** câtul împărţirii lui **a** la **b**.

```
citeşte n {n∈N,n≥2}
k←1; d←2
┌repetă
│  p←0
│ ┌cât timp n mod d=0 execută
│ │  p←p+1 ; n←n div d
│ └■
│ ┌dacă p mod 2=1 atunci k←0
│ └■
│  d←d+1
└până când (n=1)
scrie k
```

1) Ce se va afişa pentru **n=1764**?

2) Scrieţi o valoare formată din **3** cifre ce poate fi citită pentru variabila **n** astfel încât algoritmul să afişeze valoarea **0**.

3) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

4) Scrieţi un program Pascal/C/C++ echivalent cu algoritmul dat pentru toate valorile posibile ale lui **n** şi care să nu conţină nicio structură repetitivă.

84. Scrieţi un program care rezolvă următoarea problemă: se citesc de la tastatură două numere naturale distincte, fiecare de maximum **7** cifre. Să se afişeze câtul şi restul împărţirii celui mai mare dintre cele două numere la cel mai mic dintre cele două numere. Dacă împărţirea nu se poate efectua, se va afişa mesajul **EROARE.**

85. Scrieţi un program care rezolvă următoarea problemă: se citesc trei numere reale pozitive. Să se afişeze mesajul **DA** dacă cel puţin unul dintre ele are proprietatea că este egal cu media geometrică a celorlalte două şi mesajul **NU** dacă nici unul dintre ele nu este egal cu media geometrică a celorlalte două.

86. Scrieţi un program care rezolvă următoarea problemă: se citeşte un număr întreg format din cel mult **8** cifre. Să se afişeze mesajul **DA** dacă toate cifrele numărului citit sunt identice; în caz contrar, să se afişeze mesajul **NU**.

Exemple:

- dacă se citeşte unul dintre numerele **777777** sau **-9999** se va afişa mesajul **DA**;
- dacă se citeşte unul dintre numerele **777767** sau **-9099** se va afişa mesajul **NU.**

87. Scrieţi un program care citeşte trei valori reale **distincte** notate **a**, **b**, **c** şi care afişează una dintre următoarele valori:

- **1** dacă **a < b < c**;
- **2** dacă **a > b > c**;
- **3** dacă **b** este cea mai mare dintre cele trei valori;
- **4** dacă **b** este cea mai mică dintre cele trei valori;

88. Scrieţi un program care citeşte de la tastatură o valoare naturală impară **n** (**1≤n≤1000**) şi apoi calculează şi afişează pe ecran, cu exact patru zecimale, valoarea expresiei:

$$1-\sqrt{2}+\sqrt{3}-\sqrt{4}+\sqrt{5}-...+\sqrt{n}$$

Exemplu: dacă se citeşte pentru **n** valoarea **3**, programul va afişa valoarea **1,3178**

89. Scrieţi un program Pascal/C/C++ care citeşte de la tastatură un număr real **a** şi afişează pe ecran, pe acelaşi rând şi separate printr-un spaţiu, două numere întregi consecutive **x** şi **y**, cu proprietatea că **x≤a<y**.

90. Scrieţi programul Pascal/C care, pentru un număr natural nenul **n** de cel mult **4** cifre, citit de la tastatură, afişează în ordine crescătoare, separate prin spaţiu, primele **n** numere pare strict pozitive divizibile cu **5**.

De exemplu, pentru **n=6** se afişează **10 20 30 40 50 60.**

91. Scrieţi un program Pasca/C/C++ care citeşte de la tastatură un număr natural nenul **a** cu cel mult **9** cifre şi afişează ultima cifră a numărului $\mathbf{a}^{2007}$.

De exemplu, pentru **a=23467** se afişează **3**.

92. Se consideră şirul **f: 1,5,2,13,10,...** generat după regula de mai jos, în care s-a notat cu **[x]** partea întreagă a numărului **x**:

$$f_n = \begin{cases} 1 & pentru\ n = 1 \\ 2 * f_{[n/2]} + 3 & pentru\ n > 1,\ n\ par \\ 2 * f_{[n/2]} & pentru\ n > 1,\ n\ impar \end{cases}$$

a) Scrieţi un program care citeşte un număr natural **x** (cu cel mult **9** cifre) apoi afişează pe ecran mesajul **DA** dacă **x** este termen al şirului dat, respectiv mesajul **NU** în caz contrar. Se va utiliza un algoritm eficient din punct de vedere al memoriei utilizate şi al timpului de executare.
Exemplu: pentru **x=10** se va afişa **DA**

b) Descrieţi metoda utilizată şi explicaţi în ce constă eficienţa ei.

93. Pentru două puncte **A**, **B** din plan, puncte date prin coordonatele lor întregi **(xa,ya)**, **(xb,yb)**, se cere să se verifice dacă punctele A şi B sunt egal depărtate de originea axelor de coordonate. Amintim că distanţa în plan dintre punctele **P1(x1,y1)** şi **P2(x2,y2)** se calculează cu ajutorul formulei $d(P_1,P_2)=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$.

Scrieţi programul Pascal/C/C++ care citeşte de la tastatură cele 4 numere întregi şi afişează pe ecran mesajul ”DA” în cazul în care A şi B se află la aceeaşi distanţă de punctul O (originea) şi afişează mesajul ”NU“ în caz contrar.

94. Pentru un număr **x** citit de la tastatură, valoare reală pozitivă cu cel mult 5 cifre la partea întreagă şi cel mult 4 zecimale, se cere să se afişeze două numere naturale **a** şi **b** astfel încât raportul **a/b** să fie egal cu **x**. Scrieţi programul **Pascal/C** corespunzător.

De exemplu, pentru **x=1.20**, o soluţie posibilă este **6 5.**

95. Scrieţi un program Pascal/C/C++ care citeşte de la tastatură un număr natural **n** cu cel mult nouă cifre şi care determină dacă există un număr natural **k** cu proprietatea că **n=1·2·3·...·k**. Dacă există un astfel de număr, programul va afişa pe ecran mesajul **DA** urmat de numărul **k**, separate printr-un spaţiu, altfel va afişa mesajul **NU** (ca în exemple).
Exemple: pentru **n=720** se va afisa „**DA 6**” pentru că **6!=720 (6!=1*2*3*4*5*6);** pentru **n=721** se va afisa „**NU**” .

96. Pentru trei numere reale **a**, **b**, **c** citite de la tastatură, se cere să se afişeze o valoare întreagă **x** astfel încât suma distanţelor de la **x** la fiecare dintre valorile **a**, **b**, **c** să fie minimă. Valoarea determinată se va afişa pe ecran. Definim distanţa (pe axă) dintre două numere **u** şi **v** prin valoarea absolută a diferenţei lor, **|u-v|**.
a) Alegeţi o metodă corectă şi eficientă de rezolvare şi explicaţi în limbaj natural (pe maximum 5-6 rânduri) metoda aleasă justificând corectitudinea şi eficienţa acesteia.

b) Scrieţi programul Pascal/ C/C++ corespunzător metodei alese.

97. Se citesc de la tastatură numărul natural **n** (**n<30000**) şi apoi **n** numere întregi având maximum **4** cifre, cel puţin una dintre aceste valori fiind pozitivă. Se cere să se determine şi să se afişeze pe ecran cea mai mică valoare pozitivă dintre cele **n** numere citite şi să se precizeze de câte ori a apărut această valoare în şirul celor **n** numere citite. Alegeţi un algoritm de rezolvare care să utilizeze eficient memoria.

a) Descrieţi pe scurt algoritmul de rezolvare, explicând în ce constă eficienţa sa.

b) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului descris.

Exemplu. Pentru **n=8** şi valorile **6 2 -3 -5 2 9 2 6**, se afişează pe ecran valorile **2 3** (cea mai mică valoare pozitivă este **2** şi apare de **trei** ori în şir).

98. Scrieţi un program Pascal/C/C++ care verifică dacă un număr natural **n** cu cel mult **9** cifre, citit de la tastatură, este un număr „fierăstrău”. Definim număr „fierăstrău” un număr cu cel puţin **3** cifre care este format numai cu ajutorul a două cifre distincte **a** şi **b** şi şi are una dintre formele $\overline{aba}$, $\overline{ababa}$, $\overline{abababa}$, $\overline{ababababa}$, cu **a>b.** Programul afişează **DA** în caz afirmativ şi **NU** în caz contrar.

Exemple:**9393939,515** sunt numere fierăstrău iar **9354539** nu este număr fierăstrău.

99. Se citeşte de la tastatură un număr natural **n** (**1≤n≤100**). Să se afişeze pe ecran al **n**-lea termen al şirului **11,22,33,44,55,66,77,88,99, 111, 222,333,444,etc.**

De exemplu, dacă **n=11** se afişează **222.**

a) Alegeţi o metodă eficientă de rezolvare, descriind în limbaj natural metoda folosită şi justificaţi eficienţa acesteia (cel mult 6 rânduri).

b) Scrieţi programul Pascal/C/C++ corespunzător metodei descrise la punctul a).

100. Scrieţi programul **Pascal/C/C++** care afişează pe ecran, separate prin virgulă, toate numerele naturale de patru cifre care au prima cifră strict mai mică decât a doua, a doua cifră strict mai mare decât a treia şi a treia cifră strict mai mică decât a patra.
Exemplu: **1201,1202,1203,...,8979,8989**

101. Pentru două numere naturale **n** şi **m** cu maximum nouă cifre fiecare, citite de la tastatură, numere care au cifrele în ordine crescătoare, se cere să se afişeze pe ecran cel mai mare număr care se poate forma cu toate cifrele numerelor **n** şi **m.** De exemplu pentru **n=35679** şi **m=123789** se va tipări **99877653321.** Alegeţi un algoritm eficient de rezolvare.

a) Descrieţi strategia de rezolvare şi justificaţi eficienţa algoritmului ales, folosind limbajul natural (5-6 rânduri).

b) Scrieţi programul **Pascal/C/C++** corespunzător.

102. Scrieţi un program eficient din punct de vedere al timpului de executare şi al spaţiului de memorie, care citeşte de la tastatură un număr natural **n** (**n≤1000**) şi afişează pe ecran cel de al **n**-lea termen al şirului:

1, 1,2,2, 1,2,3,3,3, 1,2,3,4,4,4,4, ...

construit astfel: prima grupă este formată din numărul **1**, a doua grupă este formată din numărul **1** urmat de numărul **2** scris de două ori, etc. Grupa a **k**-a, este formată din numerele **1**, **2**,..., **k-1** urmate de numărul **k** scris de **k** ori.

103. Se citeşte de la tastatură un număr natural **n** (**n≤100**) şi apoi **n** numere din mulţimea **{1**, **2**, **3}**. Se cere să se afişeze cele **n** valori citite în ordine crescătoare.

a) Descrieţi pe scurt un algoritm de rezolvare al problemei, eficient din punct de vedere al spaţiului de memorie utilizat şi al timpului de executare, explicând în ce constă eficienţa metodei.

b) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului descris.

Exemplu. Pentru **n=19** şi valorile **3 3 3 3 2 1 2 1 3 2 1 3 2 1 1 3 3 2 3** se va afişa pe ecran şirul :
1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 3 3 3 3.

104. Să se scrie un program **Pascal/C/C++** care citeşte de la tastatură un număr natural **n** (cu cel puţin **2** cifre şi cel mult **9** cifre, toate nenule) şi care determină şi scrie pe ecran toate numerele obţinute din **n** prin eliminarea câte unei singure cifre, în ordine: mai întâi cifra unităţilor, apoi cifra zecilor, apoi cifra sutelor, etc. Numerele vor fi scrise pe o singură linie, două numere alăturate fiind separate printr-un singur spaţiu.

Exemplu: dacă se citeşte **N=12345,** se vor afişa, în ordine, numerele:

1234 1235 1245 1345 2345

105. Şirul lui Fibonacci este definit astfel:

$t_0=0,\ t_1=1,\ t_n=t_{n-1}+t_{n-2}$. pentru $n\geq 2$.

Scrieţi programul care citeşte de la tastatură două numere naturale **a>b** (**1≤a≤b≤32000**) şi verifică dacă aceste numere sunt termeni consecutivi în şirul lui Fibonacci. Se va afişa pe ecran un mesaj dacă **a** şi **b** îndeplinesc sau nu, condiţiile din enunţ. **Exemplu**: pentru **a=34, b=55** se va afişa pe ecran mesajul: **sunt termeni consecutivi in sirul lui Fibonacci**

106. Scrieţi un program Pascal/C/C++ care să afişeze pe ecran toate numerele pare de **două** cifre. Numerele vor fi scrise pe mai multe linii, câte **cinci** pe fiecare linie. Numerele de pe fiecare linie vor fi separate printr-un spaţiu.

107. Scrieţi programul Pascal/C/C++ care să afişeze toate numerele impare de două cifre. Afişarea se va face pe ecran, câte două numere pe fiecare linie, numerele de pe aceeaşi linie fiind separate printr-un spaţiu.

108. Pentru un număr natural **n** (**1<n<10.000**) citit de la tastatură să se scrie un program care determină în mod eficient toate perechile **(x,y)** de numere naturale cu proprietatea că $x^2+n=y^2$. Perechile vor fi afişate pe ecran, câte una pe linie, având valorile separate printr-un spaţiu, iar dacă nu se găseşte nicio astfel de pereche se va afişa mesajul „**nu există**”.

a) Descrieţi în limbaj natural (4-5 rânduri) metoda utilizată.

b) Explicaţi în ce constă eficienţa metodei alese (1-2 rânduri).

c) Scrieţi programul Pascal/C/C++ corespunzător.

Pentru o rezolvare mai puţin eficientă se va acorda un punctaj parţial.

109. Se citesc de la tastatură două numere naturale **n** şi **p (1<n<1000,1<p<10)**. Să se afişeze pe ecran, cu spaţiu între ele, acele numere naturale mai mici sau egale cu **n** care au toate cifrele mai mici sau egale cu **p**.
De exemplu, dacă **n=15** şi **p=2** , se vor afişa : **0 1 2 10 11 12**

110. Se citesc două numere naturale nenule cu maximum 9 cifre **a,b**. Să se verifice dacă cele două variabile sunt doi termeni consecutivi ai şirului Fibonacci. Şirul lui Fibonacci are următoarea definiţie:

$$f_1=1,\ f_2=1,\ f_i=f_{i-1}+f_{i-2}, \text{dacă } i>2$$

De exemplu, dacă **a=3** şi **b=5** atunci se va afişa mesajul **„Da”,** iar pentru **a=21** şi **b=5** se va afişa mesajul **„Nu”**

111. Pentru două valori **a** şi **b** (numere naturale mai mici decât **1000**, cu **a<b**), citite de la tastatură, se cere să se afişeze valorile întregi din intervalul închis **[a,b]** ce au cifra de control egală cu cifra de control a numărului **a**. Cifra de control a unui număr se obţine astfel: se calculează suma cifrelor numărului, apoi se calculează suma cifrelor rezultatului obţinut etc. până se obţine un număr format dintr-o singură cifră.

Exemplu: cifra de control a numărului **39** este **3** pentru că **3+9=12** iar **1+2=3**.

Alegeţi un algoritm eficient de rezolvare.

a) Explicaţi în limbaj natural metoda utilizată, justificând eficienţa ei (4-6 rânduri).

b) Scrieţi programul **Pascal/C/C++** ce rezolvă problema enunţată prin utilizarea metodei prezentate.

112. Se citesc **n** numere naturale mai mici decât **1000** (**1<n<100**). Scrieţi un program **Pascal/C/C++** ce determină ultima cifră a numărului obţinut prin adunarea tuturor elementelor citite.

Exemplu: dacă **n=3** iar numerele sunt: **102 27 34** se va afişa **3**.

113. Se consideră următorul şir de numere naturale: **1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5, ...** Primul element este **1** şi se află pe poziţia **1** în şir. Un număr **x** apare de **x** ori în şir pe poziţii consecutive şi elementele şirului sunt în ordine crescătoare.

Pentru o valoare naturală **n** (**0<n<10000**) citită de la tastatură se cere să se precizeze care este termenul cu numărul de ordine **n** din şirul prezentat.

Exemplu: dacă **n=7** atunci al şaptelea număr din şir este **4**.

Alegeţi un algoritm eficient de rezolvare.

a) Explicaţi în limbaj natural metoda utilizată, justificând eficienţa ei (4-6 rânduri).

b) Scrieţi programul **Pascal/C/C++** ce rezolvă problema enunţată, utilizând metoda descrisă la punctul anterior.

114. Se citesc de la tastatură două numere naturale **n** şi **k** (**0<n<1000000000**; **0<k<10**). Scrieţi programul **Pascal/C/C++** ce afişează numărul obţinut prin eliminarea primelor **k** cifre ale numărului **n**. Dacă numărul de cifre ce trebuie eliminate este mai mare decât numărul de cifre ale lui **n** atunci se va afişa mesajul **"NUMAR VID"**.

Exemplu: Pentru **n=1572** şi **k=2** programul afişează **72**.

115. Scrieţi un program **Pascal/C/C++** care citeşte de la tastatură un număr natural **n** (**0<n<=25**) şi un şir de **n** numere naturale cu cel mult patru cifre fiecare; programul va determina şi va afişa pe ecran numărul maxim de factori primi distincţi care pot să apară în descompunerea în factori primi a unui număr din şir.

Exemplu: pentru **n=6** şi numerele **18 90 450 24 20 75** se afişează **3** deoarece $90=2*3^2*5$ şi $450=2*5^2*3^2$, fiecare conţinând trei factori primi, iar celelalte numere citite au câte doi factori primi distincţi.

116. Scrieţi un program **Pascal/C/C++** care citeşte de la tastatură un număr natural **n** (1<**n**<30000) şi care determină şi afişează pe ecran suma exponenţilor factorilor primi care apar în descompunerea lui.

Exemplu: pentru **n=24**, se afişează **4**. (**$24=2^3*3^1$**, suma exponenţilor este **3+1=4**).

117. Scrieţi un program **Pascal/C/C++** care citeşte de la tastatură un număr natural **n** (**1<n<30000**) şi care determină şi afişează pe ecran numărul divizorilor primi ai lui **n**.
Exemplu: pentru **n=24** se afişează **2** (deoarece are doi divizori primi: **2** şi **3**), iar pentru numărul **60** se afişează **3** (deoarece are trei divizori primi: **2**, **3** şi **5**).

118. Se citeşte de la tastatură un număr natural **n** şi apoi **n** numere întregi. Să se determine cea mai mare valoare negativă dintre cele citite, precum şi numărul de apariţii ale acestei valori printre numerele citite. Alegeţi o metodă eficientă atât din punctul de vedere al timpului de executare cât şi al gestionării memoriei.

a) Descrieţi pe scurt metoda folosită explicând eficienţa acesteia (3-4 rânduri).

b) Scrieţi programul **C/C++** corespunzător metodei descrise la punctul a). Se va afişa cea mai mare valoarea negativă şi numărul de apariţii sau mesajul **NU EXISTA** dacă nu există valori negative în şirul de numere citite.

De exemplu, pentru **n=8** şi numerele **10, 3, -4, -7, -4, -7, 11, -4** se vor afişa numerele **-4** şi **3**, deoarece **-4** este cea mai mare valoare negativă şi apare de **3** ori.

119. Se citesc de la tastatură mai multe numere naturale de cel mult **3** cifre. Se ştie că se citesc cel puţin două numere şi că citirea continuă, număr cu număr, până când se citeşte un număr egal cu primul număr citit. Să se scrie un program **C/C++** care citeşte numerele şi afişează pe ecran media aritmetică a numerelor pare citite precum şi numărul acestora. Se va afişa mesajul **NU EXISTA** dacă nu există nici un număr par printre cele citite. Alegeţi o metodă de rezolvare care va gestiona eficient memoria.

De exemplu, dacă se citesc pe rând numerele **5** 6 2 7 9 6 8 **5** se vor afişa pe ecran valorile **5.5** şi **4**.

120. Scrieţi un program Pascal/C/C++ care citeşte de la tastatură **10** numere întregi, determină şi afişează pe ecran media aritmetică a tuturor valorilor pozitive, cu două zecimale exacte.

121. Scrieţi un program Pascal/C/C++ care citeşte de la tastatură două numere naturale **a** şi **b** cu acelaşi număr de cifre **(0<a<30000, 0<b<30000)**. Programul va construi şi afişa pe ecran numărul natural **c** cu proprietatea că fiecare cifră a acestuia este partea întreagă a mediei aritmetice a cifrelor situate pe aceleaşi poziţii în scrierea numerelor **a** şi **b**.

Exemplu: Pentru **a=3534** şi **b=5726**, se va afişa valoarea **c=4625**.

122. Scrieţi un program Pascal/C/C++ care citeşte de la tastatură un număr natural **n (0<n<1000)**. Să se afişeze pe ecran toate numerele naturale perfecte mai mici decât **n** separate printr-un spaţiu. Un număr natural se numeşte număr perfect dacă este egal cu suma divizorilor săi, divizori din care se exclude divizorul egal cu numărul însuşi (**6=1+2+3).**

Ex. : pentru **n=50** se va afişa **6 28**

123. Scrieţi un program **Pascal/C** care citeşte de la tastatură un număr natural **n (0 <= n <=500000)**, apoi **n** numere naturale de cel mult 9 cifre şi afişează pe ecran câte dintre aceste numere au proprietatea că sunt divizibile cu suma cifrelor lor. Pentru **n = 8** şi numerele **12, 13, 20, 51, 60, 122, 123, 117** se va afişa **4**

124. Scrieţi un program **Pascal/C** care citeşte de la tastatură un număr natural **n(0<n<101)**, apoi, de pe următoarele **n** linii, citeşte câte o pereche de numere întregi, separate prin spaţiu. Fiecare pereche **a b**, cu **a<0< b**, reprezintă extremităţile unui interval închis de forma **[a,b]** pe axa **Ox**. Afişaţi pe ecran extremităţile intervalului care reprezintă reuniunea celor **n** intervale.

De exemplu pentru **n = 3** şi perechile de numere **-1 2**
-3 8
-7 4

se va afişa **-7 8**

125. Se citeşte un număr natural **k** ($1 \le k \le 4$). Să se afişeze toate **palindroamele de 5 cifre** (numere naturale de forma $c_1c_2c_3c_4c_5$ pentru care $c_1=c_5$ şi $c_2=c_4$), cu proprietatea că diferenţa în valoare absolută dintre oricare două cifre alăturate este egală cu **k** ($|c_i-c_{i+1}|=k$, pentru orice $1 \le i \le 4$). De exemplu, pentru k=2, un astfel de număr este **53135**.

a) Descrieţi în limbaj natural o metodă de rezolvare a problemei, eficientă din punct de vedere al timpului de executare (**5-6** rânduri).

b) Scrieţi programul **Pascal/C** corespunzător metodei descrise.

126. Pentru un număr natural **n** dat, **0<n<1000000**, se cere să se verifice dacă există cel puţin un număr natural strict mai mare decât **n** care să aibă exact aceleaşi cifre ca şi **n**.

a) Descrieţi metoda de rezolvare aleasă folosind limbajul natural (3-4 rânduri)

b) Scrieţi programul **Pascal/C** care citeşte de la tastatură valoarea lui **n** şi afişează pe ecran unul dintre mesajele **EXISTA** sau **NU EXISTA**.

De exemplu, pentru **n=165**, se afişează mesajul **EXISTA**, iar pentru **n=81**, se afişează mesajul **NU EXISTA**.

127. Pentru orice număr natural nenul **n** definim **n** factorial, notat **n!**, ca fiind produsul tuturor numerelor naturale nenule mai mici sau egale cu **n (n!=1*2*...*n).**
De exemplu: **3!=1*2*3=6, 5!=1*2*3*4*5=120**

a) Descrieţi în limbaj natural o metodă eficientă de determinare a numărului de cifre nule aflate pe ultimele poziţii consecutive ale valorii obţinute în urma evaluării lui **n!**, **n** fiind un numar natural nenul de cel mult **4** cifre.
De exemplu, dacă **n=10**, cum **10!=3628800**, rezultatul va fi **2**, deoarece **3628800** are doi de **0** la sfârşit.

b) Scrieţi un program Pascal/C/C++ corespunzător metodei descrise la punctul anterior, care citeşte de la tastatură un număr natural nenul **n** de cel mult **4** cifre şi afişează pe ecran numărul de cifre nule aflate la sfarsitul lui **n!**.

128. Se citesc de la tastatură trei numere naturale de cel mult **4** cifre fiecare **n,a** şi **b** şi se cere să se afişeze pe ecran câte numere naturale mai mici sau egale cu **n** sunt multipli ai lui **a**, dar nu sunt multipli ai lui **b**. De exemplu, dacă **n=100, a=12, b=8**, rezultatul afişat va fi **4** (numerele mai mici sau egale cu **100** care sunt multipli ai lui **12** dar nu şi ai lui **8** sunt **12,36,60,84**)
a) Alegeţi o metodă eficientă de rezolvare şi descrieţi în limbaj natural metoda aleasă justificând eficienţa acesteia (4-6 rânduri).
b) Scrieţi programul Pascal/C/C++ corespunzător metodei alese la punctul a).

129. Scrieţi un program C/C++ care citeşte de pe prima linie a fişierului text **BAC.TXT** trei numere naturale nenule **a,b,c** formate din cel mult patru cifre fiecare, separate prin câte un spaţiu şi afişează pe ecran cel mai mare divizor comun al acestora. De exemplu, dacă din fişier se citesc numerele: **9 27 15**, atunci se afişează **3**.

130. Scrieţi un program Pasca/C/C++ care citeşte de la tastatură trei numere naturale **x, y şi k**, (**1≤x<y≤2000000, k≤1000)** şi afişează pe ecran cele mai mari **k** numere prime din intervalul **[x,y],** pe una sau mai multe linii separate printr-un singur spaţiu**.** Dacă nu există **k** numere prime în intervalul **[x,y]** se vor afişa toate numerele prime găsite iar pe linia următoare se va afişa mesajul "**s-au găsit mai puţine numere prime:** " urmat de numărul acestora.
De exemplu, pentru **x=3, y=12** şi **k=5** se vor afişa pe ecran nu neapărat în această ordine:
3 5 7 11
s-au găsit mai puţine numere prime:4

131. International Standard Book Number (ISBN) este un cod unic utilizat pentru identificarea fiecărei cărţi, format din **9** cifre urmate de o "cifră de control" care poate fi o cifră sau caracterul **x** ce reprezintă numărul **10**. Semnificaţia cifrelor din cadrului unui cod ISBN de forma $\overline{a_1 a_{2\ldots} a_{10}}$ este următoarea: $\overline{a_1 a_2 a_3}$ identifică ţara, $\overline{a_4 a_5 a_6 a_7}$ identifică editura, $\overline{a_8 a_9}$ identifică numărul asociat cărţii iar a_{10} reprezintă cifra de control calculată astfel încât $\sum_{i=1}^{10} a[i]*i$ să fie divizibilă cu **11**. De exemplu **973 8934 05 2** este un cod ISBN cu cifra de control **2**. Scrieţi un program Pascal/C/C++ care citeşte de la tastatură un număr natural cu **9** cifre care reprezintă primele **9** poziţii ale unui cod ISBN şi afişează pe ecran cifra de control corespunzătoare.

132. a) Scrieţi un program în limbajul Pascal/C/C++, eficient din punctul de vedere al duratei de executare, care afişează toate numerele naturale formate din cifre identice, mai mari decât **10** şi mai mici decât o valoare dată **n** , **n<= 2.000.000.000**.
De exemplu pentru **n=195** , se afişează: **11,22,33,44,55,66,77,88,99,111**.
b) Explicaţi în limbaj natural metoda utilizată, justificând eficienţa ei (4-6 rânduri).

133. Într-un sistem de coordonate carteziene se consideră punctele: **A(x_1,y_1) şi B(x_2,y_2)** unde coordonatele întregi **x_1,y_1,x_2,y_2** sunt, în această ordine, următorii termeni consecutivi ai şirului Fibonacci: **f_n,f_{n+1},f_{n+2},f_{n+3}** (**n** natural). Scrieţi un program Pascal/C/C++ care citeşte de la tastatură un număr natural **n(1<=n<=20)**, determină şi afişează pe ecran, cu 3 zecimale, lungimea segmentului **AB**. Distanţa dintre două puncte **A(x_1,y_1)** şi **B(x_2,y_2)** este $\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$ **.**

134. Scrieţi un program Pascal/C/C++ care citeşte de la tastatură două numere naturale nenule **n** şi **m** (**n<100, m<100**), determină şi afişează pe ecran toate perechile **a,b** de numere naturale nenule distincte, cel mult egale cu **n**, care au cel mai mic multiplu comun egal cu **m**. Perechile se vor afişa pe câte o linie, iar numerele corespunzătoare fiecărei perechi vor fi separate printr-un spaţiu, ca în exemplu.

De exemplu, pentru **n=6** şi **m=6** se afişează perechile:

1 6

2 3

2 6

3 6

135. Scrieţi programul Pascal/C/C++ care citeşte de la tastatură un număr natural **n (n<100)** şi un şir cu **n** numere întregi din intervalul **[100,999];** programul construieşte în mod eficient din punctul de vedere al spaţiului de memorie folosit, un şir de numere rezultat prin înlocuirea fiecărui număr din şirul citit cu numărul obţinut prin interschimbarea cifrei unităţilor cu cifra sutelor. Numerele din noul şir se vor afişa pe ecran separate printr-un singur spaţiu.
De exemplu, pentru **n=3** şi şirul **123 904 500,** se afişează: **321 409 5.**

2. Tipuri de date structurate

2.1. Tipul tablou unidimensional

2.1.1. Teste grilă (limbaj pseudocod, Pascal | C/C++)

1. Se consideră un vector cu **n** elemente reale sortate. Dacă se doreşte aflarea valorii maxime din vector, numărul minim de comparaţii necesare este:

a. **n** **b.** **2n** **c.** **1** **d.** **n-1**

2. Numărul de interschimbări care se efectuează în cazul sortării crescătoare a şirului de numere: **10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0** prin metoda bulelor este:

a. **121** **b.** **11** **c.** **10** **d.** **55**

3. De câte ori se parcurge şirul în cazul sortării descrescătoare a vectorului: **10**, **9**, **8**, **7**, **6**, **5**, **4**, **3**, **2**, **1**, **0** prin metoda bulelor?

a. o dată **b.** de **9** ori **c.** de **2** ori **d.** de **10** ori

4. Pentru căutarea unei valori între elementele unui vector ordonat descrescător, vom utiliza un algoritm eficient de tip:

a. interschimbare **b.** quick sort **c.** căutare binară **d.** backtracking

5. Cea mai mare dintre valorile strict negative memorate într-un vector **NU** există dacă:

a. în vector sunt memorate numai valori strict negative
b. în vector există cel puţin o valoare negativă
c. în vector sunt memorate valori nenule pozitive şi negative
d. în vector sunt memorate numai valori strict pozitive

6. Dacă se caută un număr **x** într-un şir de numere ordonat descrescător, căutarea secvenţială este o metodă

a. incorectă şi ineficentă **c.** corectă şi eficientă
b. corectă, dar ineficientă **d.** incorectă

7. Vectorul **A** conţine, începând cu indicele **1** elementele **(1,2,2,3,3,3,4,4,4,4,...)**. Care este valoarea elementului cu indicele **25** ?

a. **25** **b.** **7** **c.** **6** **d.** **12**

8. Pentru a verifica dacă toate elementele unui vector **A** sunt strict mai mici decât toate elementele unui alt vector **B** ordonat crescător, se compară pe rând fiecare element din **A** cu toate elementele din **B**. Această metodă de verificare este:

a. incorectă **b.** corectă dar neeficientă
c. corectă numai pentru componente întregi **d.** corectă şi eficientă

9. Pentru a verifica dacă toate elementele unui vector ordonat descrescător **A** sunt strict mai mici decât toate elementele unui alt vector **B** ordonat crescător, se compară primul element din **A** cu primul element din **B**. Această metodă de verificare este:

a. corectă numai pentru componente întregi

b. corectă şi neeficientă

c. corectă şi eficientă

d. incorectă

10. Care din secvenţele de program pseudocod de mai jos elimină corect elementul x_1 din vectorul având componentele $x_1, x_2, \ldots, x_n$?

a.
```
i←1
┌cât timp i<=n-2 execută
│ xi←xi+1; i←i+1
└■
n←n-1
```

b.
```
i←2
┌cât timp i<=n execută
│ xi←xi-1; i←i+1
└■
n←n-1
```

c.
```
i←2
┌cât timp i<=n execută
│ xi-1←xi; i←i+1
└■
n←n-1
```

d.
```
i←2
┌cât timp i<=n execută
│ xi←xi+1; i←i+1
└■
n←n-1
```

11. Pentru un vector **v** cu **101** componente reale se utilizează secvenţa pseudocod alăturată pentru a verifica dacă orice două elemente distincte egal depărtate de capetele vectorului au valori diferite. Cu ce expresie logică trebuie înlocuite punctele de suspensie?

```
i←1; j←101
cât timp ai≠aj execută
  i←i+1; j←j-1
sfârşit cât timp
dacă ... atunci scrie 'DA'
         altfel scrie 'NU'
sfârşit dacă
```

a. $a_i=a_j$ **b.** $a_i\neq a_j$ **c.** $i\neq j$ **d.** $i=j$

12. Pentru a sorta vectorul **v**, care are **100** de componente numerotate de la **1** la **100**, se foloseşte algoritmul pseudocod alăturat. Care este numărul maxim de interschimbări care pot fi executate?

```
┌pentru i=1,99 execută
│ x←i
│ ┌pentru j=i+1,100 execută
│ │┌dacă vj<vx atunci
│ ││ x←j
│ │└■
│ └■
│ ┌dacă x≠i atunci
│ │ t←vx; vx←vi; vi←t
│ └■
└■
```

a. **50** de interschimbări

b. **200** de interschimbări

c. **99** de interschimbări

d. **100** de interschimbări

13. Fie **v** un vector cu **n=9** componente întregi.
Ce va afişa secvenţa alăturată pentru **v =(14, 3, 7, 0, -4, 3, 10, 15, 7)** ?

S-a notat cu **x mod y** restul împărţirii numărului natural **x** la numărul natural **y**.

```
s←0
┌pentru i=0, n-1 execută
│   ┌dacă (i mod 2 =0)
│   │        atunci s←s+v_i
│   └■
└■
scrie s
```

a. 34 **b.** 21 **c.** 50 **d.** 0

14. Algoritmul alăturat determină în variabila **K** numărul tuturor componentelor ce memorează o valoare pozitivă de cel puţin trei cifre dintre cele **15** componente întregi ale vectorului **a**. Cu ce pot fi înlocuite punctele de suspensie?

```
K←15
┌pentru i←1,15 execută
│ ┌dacă 99 >= ... atunci
│ │ K←...+K
│ └■
└■
```

a. a_i şi **1** **b.** a_i şi **-i** **c.** a_i şi **-1** **d.** **K** şi **-1**

15. Algoritmul alăturat determină în variabila **K** suma tuturor componentelor pozitive dintre cele **15** componente întregi ale vectorului **a**. Cu ce pot fi înlocuite punctele de suspensie?

```
K←0
┌pentru i←1,15 execută
│ ┌dacă a_i>=...
│ │   atunci K←...+K
│ └■
└■
```

a. **0 şi a_1** **b.** **0 şi a_i** **c.** **K şi a_i** **d.** **a_1 şi a_i**

16. În secvenţa alăturată s-a notat cu **x%y** restul împărţirii numărului întreg **x** la numărul întreg **y**. Această secvenţă are ca efect:

```
┌pentru i ← 1, n execută
│┌dacă (i%2=0) şi (a[i]%2>0)
│|     atunci  scrie a[i]
│└■
└■
```

a. Tipărirea elementelor de valoare impară aflate pe poziţii pare în tabloul **a**
b. Tipărirea elementelor de valoare impară aflate pe poziţii impare în tabloul **a**
c. Tipărirea elementelor de valoare pară aflate pe poziţii impare în tabloul **a**
d. Tipărirea elementelor de valoare pară aflate pe poziţii pare în tabloul **a**

17. Fie **v** un vector cu **n=9** componente întregi.
Ce va afişa secvenţa alăturată pentru **v=(14,3,7,0,-4, 3,10,15,7)** ?

```
s←0
┌pentru i=0, n-1 execută
│   ┌dacă (i mod 2 =0)
│   │        atunci s←s+v_i
│   └■
└■
scrie s
```

a. 34 **b.** 21 **c.** 50 **d.** 0

18. Fie tabloul unidimensional **a** în care elementele sunt, în ordine: **1,3,5,7,10,16,21**. Pentru a verifica dacă numărul **x=4** se află printre elementele tabloului, se aplică metoda căutării binare. Care este succesiunea corectă de elemente cu care se compară **x** ?

a. 1,3,5 **b.** 7,5,3 **c.** 7,3,5 **d.** 21,16,10,7,5,3

19. Căutarea unui element într-un vector ordonat descrescător se realizează în mod eficient cu un algoritm care utilizează:

a. metoda căutării binare
b. sortarea crescătoare a vectorului
c. metoda backtracking
d. parcurgerea iterativă a vectorului

20. Fie trei secvenţe de numere:

i) 1,4,6,8,9 ii) 8,5,4,3,2,1 iii) 2,3,8,5,9

Algoritmul de căutare binară se poate aplica direct, fără alte prelucrări prealabile:

a. numai secvenţei **i**
b. numai secvenţei **iii**
c. numai secvenţei **ii**
d. atât secvenţei **i** cât şi secvenţei **ii**

21. Se consideră două tablouri unidimensionale **A** şi **B**: **A=(1,3,5,9,10)** respectiv **B=(2,4,6,7)**. In urma interclasării lor în ordine crescătoare se obţine tabloul cu elementele:

a. (1,2,3,4,5,6,9,7,10)
b. (1,2,3,4,5,6,7,9,10)
c. nu se poate realiza interclasarea
d. (1,3,5,9,10,2,4,6,7)

22. Se consideră o stivă implementată prin intermediul vectorului **a** cu elementele : **a[0]=0, a[1]=10, a[2]=20, a[3]=30, a[4]=40, a[5]=50**. Dacă cel de-al doilea element, începând de la baza stivei, este **10**, atunci primul element care iese din stivă este:

a. a[6] **b.** a[1] **c.** a[5] **d.** a[0]

23. Un program care urmează să sorteze crescător **15** numere reale aflate într-un fişier, citeşte aceste numere într-o variabilă **x**. Cum trebuie declarată această variabilă?

a. `var x:integer;`	**a.** `int x;`
b. `var x:array[1..15]of real;`	**b.** `float x[15];`
c. `var x:array[1..15]of word;`	**c.** `unsigned x[15];`
d. `var x:real;`	**d.** `float x;`

24. Într-o variabilă se vor memora simultan cele **18** medii semestriale la disciplinele studiate şi media aritmetică a acestora. Care este declararea corectă pentru acestă variabilă?

a. `var a:array[1..18] of real;`	**a.** `double a[18];`
b. `var a:array[1..19] of real;`	**b.** `float a[19];`

c.	`var a:array[1..19] of byte;`	`int a[19];`
d.	`var a:array[1..20] of word;`	`long a[20];`

25. Care este declararea corectă a unei variabile **x** pentru a memora **3** numere întregi împreună cu media lor aritmetică?

a.	`var x:array[1..4] of real;`	`float x[4];`
b.	`var x: real[1..4];`	`int x[3];`
c.	`var x::array[4] of integer;`	`float x[3];`
d.	`var x:array[1..4] of integer;`	`int x[4];`

26. Care dintre următoarele declarări **nu** poate fi o declarare corectă a unei variabile **a** utilizată pentru memorarea simultană a cel mult **100** de numere reale?

a.	`var a:record`
	`  x,y:array[1..50] of real;`
	`end;`
		`struct {float c[50];`
		`  float b[50];`
		`} a;`

b.	`var a:array[1..20,1..5] of real;`	`float a[20][5];`
c.	`var a:array[1..100] of real;`	`float a[100];`
d.	`var a:array[1..100] of integer;`	`int a[99];`

27. Variabilele **a** şi **i** sunt declarate astfel:

```
Var i:integer;
    a:array[1..100] of integer;
```

```
int a[100], i;
```

Care dintre următoarele instrucţiuni este corectă din punct de vedere sintactic?

a.	`a[1] := i`	`a[1] = i;`
b.	`a :=  i`	`a =  i;`
c.	`i :=  a`	`i =  a;`
d.	`i[1] := a`	`i[1] = a;`

28. În secvenţa alăturată variabilele **i** şi **n** sunt de tip **integer/int**, **min** este de tip **real/float**, iar vectorul **x**, cu indicii de la **1**, conţine **n** numere reale. Cu ce trebuie înlocuite punctele de suspensie din secvenţă astfel încât, la finalul executării ei, variabila **min** să conţină cea mai mică valoare memorată de componentele vectorului **x** ?

```
min:=....
for i:=2 to n do
 if .... then min:=x[i];
```

```
min= ....
for(i=2; i<=n; i++)
  if (....) min=x[i];
```

a.	`0;` şi `x[i]<min`	`0;` şi `x[i]<min`
b.	`x[1];` şi `x[i]<min`	`x[1];` şi `x[i]<min`
c.	`x[1];` şi `x[i] >=min`	`x[1];` şi `x[i]>=min`
d.	`x[n];` şi `x[i]<=min`	`x[n];` şi `x[i]<=min`

29. Fie vectorii **a** şi **b** în care **a =(1,2,4,5,3)** şi **b=(4,2,1,3)**. Atunci **a[b[1]]** are valoarea:

a. 5 **b.** 3 **c.** 2 **d.** 1

30. Se consideră tabloul unidimensional **a=(0,1,3,2).** Indicaţi tabloul unidimensional **b** astfel încât pentru orice **0<=i<4** să existe relaţia **a[b[i]]=b[a[i]].**

a. b=(0,1,2,3) **b.** b=(3,1,0,2) **c.** b=(2,3,0,1) **d.** b=(1,0,2,3)

31. Fie **v** un vector cu **n** elemente de tip întreg, iar **n** un număr natural nenul **(n<=100).** De câte ori se repetă instrucţiunea **i:=i+1; / i++;** în timpul executării secvenţei de instrucţiuni următoare?

```
i:=0;
while i<n do
  begin
     i:=i+1; v[i]:=i*i
  end
```

```
i=0;
while (i<n)
  { i++;
    v[i]=i*i;
  }
```

a. n ori **b.** n+1 ori **c.** 0 ori **d.** n-1 ori

32. Fie **v** un tablou unidimensional format din **n** componente întregi, numerotate de la **1** la **n**, iar **p** şi **k** două variabile de tip întreg. În care dintre următoarele secvenţe variabilei **p** i se atribuie prima valoare strict pozitivă din tabloul **v**, dacă există o astfel de valoare, sau **0** in caz contrar?

a.
```
p:=0;k:=1;
while v[k]>=0 do k:=k+1;
p:=v[k]
```

b.
```
p:=0;k:=n;
while v[k]<=0 do k:=k-1;
p:=v[k]
```

c.
```
p:=0;
for k:=1 to n do
  if (v[k]>0)and(p=0)
      then p:=v[k]
```

d.
```
p:=0;
for k:=1 to n do
  if v[k]>0 then p:=v[k]
```

a.
```
p=0;k=1;
while(v[k]>=0) k++;
p=v[k];
```

b.
```
p=0;k=n;
while(v[k]<=0) k--;
p=v[k];
```

c.
```
p=0;
for(k=1;k<=n;k++)
  if((v[k]>0)&&(p==0))
     p=v[k];
```

d.
```
p=0;
for(k=1;k<=n;k++)
  if(v[k]>0) p=v[k];
```

33. Fie **v** un tablou unidimensional format din **n** numere întregi (**n>=2**). Ştiind că primul element din tablou se află pe poziţia **1**, indicaţi care dintre următoarele secvenţe atribuie variabilei întregi **max** cea mai mare valoare din tabloul **v**:

a.
```
for i:=2 to n do
 if v[i-1]<=v[i] then
    v[i-1]:=v[i];
write(v[1])
```

a.
```
for(i=2;i<=n;i++)
  if(v[i-1]<=v[i])
      v[i-1]=v[i];
max=v[1];
```

b.
```
for i:=1 to n-1 do
 if v[i+1]>v[i] then
    v[i]:=v[i+1];
write(v[1])
```

c.
```
for i:=2 to n do
 if v[i-1]>v[i] then
    v[i]:=v[i-1];
write(v[n])
```

d.
```
for i:=n downto 2 do
 if v[i]>v[i-1] then
    v[i]:=v[i-1];
write(v[n])
```

b.
```
for(i=1;i<=n-1;i++)
  if(v[i+1]>v[i])
      v[i]=v[i+1];
max=v[1];
```

c.
```
for(i=2;i<=n;i++)
  if(v[i-1]>v[i])
      v[i]=v[i-1];
max=v[n];
```

d.
```
for(i=n;i>=2;i--)
  if(v[i]>v[i-1])
      v[i]=v[i-1];
max=v[n];
```

34. Se considera tabloul unidimensional **a=(1,2,3,4,5)**. Care va fi rezultatul afişat în urma executării secvenţei următoare?

```
for i:=1 to 5 do begin
  s:=0;
  for j:=1 to i do
         s=s+a[i];
  write(s)
end
```

```
for (i=0;i<5;i++)
{s=0;
 for(j=0;j<=i;j++) s=s+a[j];
 printf(„%d”,s); /cout<<s;
}
```

a. 1151515 **b.** 15 **c.** 6101315 **d.** 1361015

35. Care dintre secvenţele de mai jos conduce la memorarea în variabila **max** a celei mai mari dintre cele **n** valori întregi reţinute în tabloul unidimensional **a** (în care **a[1]** este primul element) ?

a.
```
for i:=1 to n-1 do
 if a[i]>a[i+1] then max:=a[i];
```

b.
```
for i:=1 to n-1 do
 if a[i]>a[i+1] then
 begin
  max:=a[i];
  a[i]:=a[i+1];
  a[i+1]:=max
end; max:=a[n];
```

c.
```
max:=a[1]; for i:=2 to n do
if a[i]<max then max:=a[i];
```

d.
```
max:=0;
for i:=1 to n do
if a[i]>max then a[i]:=max;
```

a.
```
for(i=0;i<n-1;i++)
  if(a[i]>a[i+1])
  max=a[i];
```

b.
```
for(i=0;i<n-1;i++)
  if(a[i]>a[i+1])
  { max=a[i];
    a[i]=a[i+1];
    a[i+1]=max;
  }
  max=a[n-1];
```

c.
```
max=a[0];
for(i=1;i<n;i++)
  if(a[i]<max) max=a[i];
```

d.
```
max=0;
for(i=0;i<n;i++)
  if(a[i]>max)a[i]=max;
```

36. Cu ce expresie trebuie înlocuite punctele de suspensie astfel încât în urma executării secvenţei alăturate să se deplaseze elementele v_q, v_{q+1}, ... v_k ale unui tablou unidimensional **v** cu **p-1** poziţii spre dreapta?

```
for j:=k downto q do
  v[.....]:=v[j]
```

```
for (j=k; j>=q; j--)
  v[.....]=v[j]
```

a. p-1-j **b.** j-p+1 **c.** p-1+j **d.** p-j+1

37. Se consideră următoarele declarări:

```
const x:vector=array[0..4] of
integer =(0,1,5,3,4);
var y,i:integer;
```

```
int  x[5]={0,1,5,3,4};
int y,i;
```

Ce va afişa secvenţa următoare?

```
y:=x[1];
for i:=0 to 4 do
   if y<x[i] then  y:=x[i];
write(y);
```

```
y=x[1];
for(i=0;i<=4;i++)
  if(y<x[i]) y=x[i];
cout<<y;
```

a. 0 **b.** 5 **c.** 13 **d.** nedeterminată

38. Se consideră un tablou unidimensional **a** în care:

```
a[1]=4, a[2]=2, a[3]=5,
a[4]=1, a[5]=3.
```

```
a[0]=4, a[1]=2, a[2]=5,
a[3]=1, a[4]=3.
```

Care va fi conţinutul său după executarea secvenţei de mai jos (în care variabilele **i** şi **aux** sunt de tip **integer/int**)?

```
for i:=1 to 4 do
 if a[i]>a[i+1] then begin
  aux:=a[i];
  a[i]:=a[i+1];a[i+1]:=aux
 end;
```

```
for (i=0;i<4;i++)
  if (a[i]>a[i+1]){
   aux=a[i];
   a[i]=a[i+1];a[i+1]=aux;
  }
```

a. a[1]=2, a[2]=4, a[3]=1, a[4]=3, a[5]=5

b. a[1]=4, a[2]=2, a[3]=5, a[4]=1, a[5]=3

c. a[1]=1, a[2]=2, a[3]=3, a[4]=4, a[5]=5

d. a[1]=4, a[2]=2, a[3]=1, a[4]=3, a[5]=5

a. a[0]=2, a[1]=4, a[2]=1, a[3]=3, a[4]=5

b. a[0]=4, a[1]=2, a[2]=5, a[3]=1, a[4]=3

c. a[0]=1, a[1]=2, a[2]=3, a[3]=4, a[4]=5

d. a[0]=4, a[1]=2, a[2]=1, a[3]=3, a[4]=5

39. Se consideră un tablou unidimensional **a** în care elementele sunt, în ordine : **1,3,5,7,10,16,21**. Pentru a afla poziţia pe care se află valoarea **x=10** se aplică metoda căutării binare. Care este succesiunea corectă de elemente a căror valoare se compară cu valoarea lui **x** ?

a. 21,16,10 **b.** 7,16,10 **c.** 1,3,5,7,10 **d.** 5,7,10

40. Dacă **n=10** şi vectorul **a** conţine, începând de la poziţia **0** până la poziţia **9** valorile **3, 5, 2 , 6, 8, 2, 1, 6, 9, 10** în aceasta ordine, ce afişează secvenţa de instrucţiuni de mai jos?

```
for i:=0 to n-1 do
  if (i mod 2=1) then
    if(a[i] mod 2=0) then
      write(a[i],' ');
```

```
for(i=0;i<n;i++)
  if (i%2==1)
    if(a[i]%2==0)
     cout<<a[i]<<" ";
    / printf("%d ",a[i]);
```

a. 2 8 **b.** 6 2 6 10 **c.** 6 2 6 **d.** 5

41. Ştiind că variabila **v** este un tablou unidimensional cu **100** de componente ce memorează valori distincte de tip **integer/int**, care este valoarea reţinută de componenta **v[1]/v[0]** în urma executării secvenţei de instrucţiuni următoare?

```
for i:=100 downto 2 do
  if v[i]<v[i-1] then begin
   v[i]:=v[i]+v[i-1];
   v[i-1]:=v[i]-v[i-1];
   v[i]:=v[i]-v[i-1];
  end;
```

```
for(i=99;i>0;i--)
  if(v[i]<v[i-1]){
   v[i]=v[i]+v[i-1];
   v[i-1]=v[i]-v[i-1];
   v[i]=v[i]-v[i-1];
  }
```

a. valoarea care apare cel mai frecvent în tablou

b. cel mai mic element al tabloului

c. suma elementelor tabloului

d. cel mai mare element al tabloului

2.1.2. Probleme (programe pseudocod, Pascal | C/C++)

1. **Se consideră programul pseudocod alăturat** în care s-a folosit notaţia **[x]** pentru partea întreagă a lui **x** .

1) Ce se va afişa pentru **c=8**?

2) Scrieţi o valoare strict pozitivă de o cifră pentru variabila **c** astfel încât dacă se citeşte această valoare, programul afişează valorile **0** şi **4**.

3) Scrieţi programul pseudocod care să fie echivalent cu algoritmul dat şi care să folosească o structură repetitivă cu test iniţial în locul structurii repetitive **repetă...până când**.

4) Scrieţi programul Pascal/C/C++ corespunzător cu algoritmul dat.

```
┌pentru i=1,10 execută
│  a_i = 2*i
└■
citeşte c     (număr întreg)
s←1 ; d←10 ; g←0 ; k←0
┌repetă
│ m← [(s+d)/2]
│ k← k+1
│┌dacă c=a_m atunci g←1
│└  altfel
│   ┌dacă c<a_m atunci
│   │    d←m-1
│   │altfel s←m+1
│   └■
└■ până când g=1 sau s>d
scrie g,k
```

2. Se consideră programul pseudocod alăturat:

```
citeşte n
      (număr natural nenul)
pentru i=1,n execută
  citeşte a_i
■
m← a_1
pentru i=2,n execută
  dacă a_i este număr
       par atunci
    dacă m < a_i atunci
      m ← a_i
    ■
  ■
■
scrie m
```

1) Ce se va afişa dacă se citesc valorile **8, 1, 7, 10, 3, 12, 1, 2, 6**?

2) Scrieţi un şir de cel puţin **6** valori care dacă este citit determină afişarea unei valori impare

3) Scrieţi programul pseudocod care să fie echivalent cu algoritmul dat şi care să folosească alte tipuri de structuri repetitive.

4) Scrieţi programul Pascal/C/C++ corespunzător cu algoritmul dat.

3. Se consideră programul pseudocod alăturat:

S-a notat cu **x%y** restul împărţirii numărului natural **x** la numărul natural **y** şi cu **[z]** partea întreagă a numărului real **z.**

```
citeşte n  {număr natural }
pentru i←0,9 execută
  a_i←0
■
repetă
 i← n%10; a_i← a_i +1;
n←[n/10]
■ până când n=0
pentru i←9,0,-1 execută
  pentru j←1,a_i
     scrie i
  ■
■
```

1) Ce se va afişa pentru **n=10220685**?

2) Câte valori pot fi introduse pentru variabila **n** astfel încât rezultatul afişat în urma executării algoritmului să fie **720**.

3) Scrieţi programul pseudocod care să fie echivalent cu algoritmul dat şi care să conţină o structură repetitivă cu test iniţial în locul structurii repetitive cu test final.

4) Scrieţi programul Pascal/C/C++ corespunzător cu algoritmul dat.

4. Se consideră programul pseudocod alăturat:

S-a notat cu a_k elementul aflat pe poziţia **k** în tabloul unidimensional cu numele **a**, cu **|x|** valoarea absolută a numărului natural **x** şi cu **[x]** partea întreagă a numărului **x.**

```
pentru i←1,5 execută
citeşte a_i (număr întreg)
■
pentru i←1,4 execută
 m←a_i+a_i+1+|a_i-a_i+1|
 a_i+1←a_i+a_i+1-[m/2]
 a_i←[m/2]
■

scrie a_5
```

1) Ce se afişează dacă sunt citite, în ordine, numerele **3, 5, 8, 1, 7?**

2) Scrieţi o succesiune de valori citite pentru elementele a_1, a_2, a_3, a_4, a_5 astfel încât să se afişeze prima valoare introdusă.

3) Explicaţi în limbaj natural **efectul** atribuirilor

$m \leftarrow a_i + a_{i+1} + |a_i - a_{i+1}|$

$a_{i+1} \leftarrow a_i + a_{i+1} - [m/2]$

$a_i \leftarrow [m/2]$

din algoritmul dat.

4) Scrieţi programul PASCAL/C/C++ corespunzător algoritmului dat alăturat.

5. **Se consideră programul pseudocod următor,** unde **v** reprezintă un tablou unidimensional cu elemente numere întregi în care v_1 este primul element al tabloului, v_2 este al doilea, ... , iar v_n este ultimul.

1) Care este valoarea afişată dacă de la tastatură se citesc, în ordine, valorile **5,1,7,3,2,3**?
2) Dacă **n=7**, iar următoarele **7** valori citite sunt egale între ele, care trebuie să fie valoarea lor astefel încât să se afişeze **8** ?
3) Scrieţi programul PASCAL/C/C++ corespunzător algoritmului dat.

```
citeşte n (n∈N impar)
┌pentru i=1,n execută
│ citeşte v_i
└■
 i←0; j←0
┌cât timp j<n execută
│ i←i+1; j←j+2
└■
scrie v_i
```

4) Scrieţi un algoritm pseudocod echivalent cu cel dat care să nu utilizeze alte structuri repetitive în afara celei folosite pentru citirea tabloului.

6. **Se consideră programul pseudocod alăturat** .S-au folosit următoarele notaţii **mod** pentru restul împărţirii întregi iar **div** pentru câtul împărţirii întregi .

1) Ce se va afişa pentru **n=20**?
2) Pentru câte valori ale lui **n** se vor afişa exact **6** numere.
3) Scrieţi un program pseudocod care să fie echivalent cu cel dat, dar în care să se înlocuiască prima structură repetitivă pentru cu o structură repetitivă cu test iniţial.
4) Scrieţi programul **PASCAL/C/C++** corespunzător algoritmului dat.

```
citeşte n {număr natural >1}
┌pentru i← 1,n execută
│  v[i] ← i
└■

┌pentru i←2,[√n ]execută
│   ┌dacă v[i]<>0 atunci
│   │ j← i
│   │┌repetă
│   ││ j←j+i;v[j] ← 0
│   │└până când j>n
│   └■
└■
┌pentru i←2,n execută
│ ┌dacă v[i]<>0 atunci
│ │   scrie i
│ └■
└■
```

7. **Se consideră programul pseudocod alăturat:**

S-a notat cu **x%y** restul împărţirii numărului natural **x** la numărul natural **y**.

```
citeşte n (n natural<100)
┌pentru i←1,n execută
│ citeşte v_i (întreg)
└■
```

1) Ce valori se vor afişa în urma executării programului pseudocod pentru **x=3** şi **v=(4,6,11,16,9,12,21,8)**?

2) Daţi o valoare variabilei **x** astfel încât pentru şirul de numere de la punctul anterior valoarea variabilei **nr** să rămână **0**.

```
citeşte x; nr ←0
┌pentru i←1,n execută
│┌dacă v_i%x=0 atunci
│└ nr←nr+1; scrie v_i
└┘
scrie nr
```

3) Ce proprietate trebuie să îndeplinească elementele vectorului **v** pentru a fi afişate?

4) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

8. Se consideră programul pseudocod alăturat:

1) Ce se va afişa în urma executării secvenţei pseudocod alăturată, dacă pentru numărul natural **n** se citeşte valoarea **8**, apoi cele opt componente citite de la tastatură sunt în ordine: **0**, **1**, **3**, **3**, **5**, **6**, **4**, **8**.

2) Pentru **n = 5** scrieţi un alt set de valori pentru componentele şirului, astfel încât variabila **s** să aibă valoarea **0**, în urma executării algoritmului.

3) Scrieţi programul **Pascal/C/C++** corespunzător algoritmului dat.

```
citeşte n (nr natural de cel
mult trei cifre)
┌pentru i←1,n execută
│ citeşte a_i (nr. natural)
└┘
s ← 0
┌pentru i←1,n execută
│  ┌dacă a_i = i atunci
│  │  a_i ← a_i + i
│  │altfel
│  │  s ← s + 1
│  └┘
└┘
scrie s
┌pentru i←1, n execută
│     scrie a_i
└┘
```

4) Înlocuiţi în secvenţa de pseudocod expresia **a_i = i** cu o altă expresie, astfel încât variabila **s** să afişeze numărul de componente pare ale şirului.

9. Se consideră programul pseudocod alăturat care prelucrează numerele naturale nenule dintr-un tablou unidimensional **a**, de dimensiune **n**. S-a notat cu **x%y** restul împărţirii numerelor întregi **x** şi **y**.

1) Pentru **n=7** şi **a=(3,7,5,49, 21,77,70)** precizaţi ce se afişează în urma executării algoritmului.

2) Pentru **n=7** daţi un exemplu de valori pentru **a,** astfel încât variabila **nr** să memoreze la finalul executării algoritmului cea mai mare valoare posibilă.

3) Pentru **n=5** scrieţi un exemplu de valori pentru **a** astfel încât rezultatul afişat în urma executării algoritmului să fie: **0 0.**

```
citeşte n
(n nr natural, 1<n≤100)
┌pentru i←1,n execută
│   citeşte a_i
└┘
max←0; nr←0;
┌pentru i←1,n execută
│┌dacă a_i%7=0 atunci
│└   nr←nr+1
│   ┌dacă a_i>max atunci
│   │      max←a_i
│   └┘
└┘
scrie max,nr
```

4) Scrieţi programul Pascal/C/C++ corespunzător algoritmului din enunţ.

10. Se consideră programul pseudocod alăturat:

```
citeşte n
      {n număr natural, 1<n≤100}
pentru i←1,n execută
   citeşte a_i
■
pentru i←2,n execută
   x← a_i; j←i-1
cât timp j>0 şi x<a_j execută
   a_{j+1}←a_j; a_j←x; j←j-1
■
■
pentru i←1,n execută
   scrie a_i
■
```

1) Care sunt valorile afişate în urma executării, dacă se citeşte succesiunea de valori: **5**, **7**, **8**, **1**, **0**, **6** ?

2) Precizaţi o succesiune de **7** valori care pot fi citite astfel încât instrucţiunile din structura repetitivă **cât timp** să nu se execute niciodată.

3) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

4) Scrieţi un program pseudocod echivalent cu algoritmul dat, care utilizează o structură repetitivă cu test final în locul structurii repetitive **cât timp**.

11. Se citeşte de la tastatură un număr natural **n**, **0<n<1000000**. Să se afişeze pe ecran, dacă există, un număr natural care este strict mai mare decât **n** şi care are exact aceleaşi cifre ca şi **n**. Dacă nu există un astfel de număr, se va afişa mesajul **Nu există**.

De exemplu, pentru **n=165**, se poate afişa valoarea **561**.

a) Alegeţi un algoritm eficient ca timp de executare. Descrieţi metoda în limbaj natural pe cel mult 4 rânduri.

b) Scrieţi programul **Pascal/C** corespunzător metodei descrise.

12. Scrieţi un program care construieşte în memorie un vector **a** cu toate numerele de două cifre, cu cifre distincte, pare, aflate în ordine descrescătoare. Elementele vectorului vor fi în ordine strict descrescătoare. Afişaţi elementele vectorului **a** pe o linie a ecranului despărţite printr-un spaţiu.

Exemplu: vectorul va avea elementele: **86, 84, 82, 80, 64, ..., 20.**

13. Scrieţi un program care citeşte de la tastatură un vector **a** cu **15** elemente numere reale pozitive şi afişează pe ecran pentru fiecare element al vectorului cel mai mic număr natural mai mare sau egal cu el şi cel mai mare număr natural mai mic sau egal cu el. Valorile vor fi afişate pe ecran despărţite printr-un spaţiu.

Exemplu: Dacă vectorul **a** conţine elementele: **12.3 1.98 14.67 1 3.11 2.08 3.9 12.89 7.99912 8 6.7 1.001 13 8.9909 4.7** pe ecran se vor afişa numerele: **13 12 2 1 15 14 1 1 4 3 3 2 4 3 13 12 8 7 8 8 7 6 2 1 13 13 9 8 5 4**

14. Se citesc de la tastatură două valori naturale **m** şi **n** (**m,n<=100**), iar apoi **m+n** numere întregi de cel mult **9** cifre fiecare. Dintre cele **m+n** numere citite primele **m** sunt ordonate strict crescător, iar următoarele n sunt deasemenea ordonate strict crescător. Se cere să se afişeze pe ecran care din cele **m+n** numere au fost citite de două ori.

a) Descrieţi un algoritm eficient de rezolvare a acestei probleme, explicând în ce constă eficienţa acestuia.

b) Scrieţi programul **Pascal** corespunzător algoritmului descris.

De exemplu pentru **m=7** şi **n=10** şi valorile **1,2,3,4,5,6,20,3,5,7,8, 9,10,12,20, 24,35** se vor afişa valorile **3 5 20**.

15. Pe o hartă sunt figurate **n** ţări, numerotate cu **1,2,...n** (**2<n<100**). Se consideră că ţara **i** este vecină cu ţara **j** dacă **i=[j/2]** sau **j=[i/2]** (s-a notat cu **[x]** partea întreagă a numărului real **x**). De exemplu, pentru **n=3**, ţările **1** şi **2** sunt vecine deoarece **1=[2/2]**, ţările **3** şi **1** sunt vecine deoarece **1=[3/2]**, în timp ce ţările **2** şi **3** nu sunt vecine deoarece **2≠[3/2]** şi **3≠[2/2]**.

Harta a fost colorată cu **4** culori (notate cu **1, 2, 3, 4**) şi se consideră că o variantă de colorare este corectă dacă oricare două ţări vecine au o culoare diferită.

Scrieţi un program **Pascal/C/C++** care citeşte **n**, apoi **n** numere despărţite prin câte un spaţiu, reprezentând, în ordine, culoarea fiecăreia dintre cele **n** ţări (de la ţara **1** la ţara **n**) şi verifică dacă harta a fost colorată corect sau nu, afişând pe ecran mesajul „**corect**”, respectiv „**incorect**”.

Exemplu: dacă se introduc valorile **n=4** şi numerele **1 2 3 2** corespunzător asocierilor: ţara **1**-culoarea **1**, ţara **2**-culoarea **2**, ţara **3**-culoarea **3**, ţara **4**-culoarea **2**, atunci se va afişa **incorect** pentru că ţările **2** şi **4** au aceeaşi culoare şi sunt vecine.

16. Se citeşte de la tastatură un şir de numere naturale mai mici decât **10000** (şirul are cel mult **100** de valori). Şirul se termină atunci când se introduce o valoare negativă (aceasta valoare nu face parte din şir). Scrieţi programul **Pascal/C/C++** ce afişează pe ecran, în ordine crescătoare, toate numerele impare din şir, separate prin câte un spaţiu.

De exemplu, dacă au fost introduse următoarele valori: **7 2 1 9 4 0 7 3 22 -3** se vor afişa valorile **1 3 7 7 9.**

17. Se citesc de la tastatură **n** (**n<100**) numere naturale mai mici decât **10000**. Scrieţi programul **Pascal/C/C++** ce afişează pe ecran în ordine crescătoare toate numerele ce încep şi se termină cu aceeaşi cifră. Numerele se afişează separate prin câte un spaţiu.

Exemplu: Dacă **n=6**, iar numerele citite sunt: **21 3123 7 454 45 10** atunci programul va afişa: **7 454 3123**.

18. Scrieţi un program Pascal/C/C++ care citeşte un număr natural nenul par, **n, n<100** şi apoi **n** numere naturale de cel mult **4** cifre fiecare şi determină cea mai mare sumă care poate fi obţinută adunând numai o jumătate din toate numerele citite. Rezultatul se va afişa pe ecran. De exemplu, pentru **n=6** şi numerele **728,10,103,44,1000,94** se va afişa: **1831** (reprezentând suma: **728+103+1000**).

19. Scrieţi un program C/C++ care citeşte de la tastatură un număr natural nenul **n** (**n≤1000**), construieşte în memorie şi apoi afişează pe ecran un tablou unidimensional **a**, având **n** elemente. Tabloul **a** se completează cu numerele naturale de la **1** la **n**, astfel: se memorează valoarea **1** pe poziţia **1**, valoarea **2** pe poziţia **n**, **3** pe poziţia **2**, **4** pe poziţia **n-1**, **5** pe poziţia **3**, etc. Elementele tabloului se afişează pe ecran cu câte un spaţiu între ele.

Exemplu: pentru **n=9** tabloul afişat este: **1 3 5 7 9 8 6 4 2**

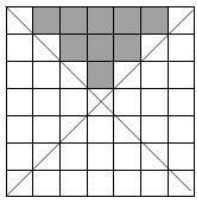
2.2. Tipul tablou bidimensional

2.2.1. Teste grilă (limbaj pseudocod, Pascal | C/C++)

1. Se consideră două tablouri unidimensionale **A** şi **B**: **A=(1,3,5,9,10)** respectiv **B=(2,4,6,7)**. In urma interclasării lor în ordine crescătoare se obţine tabloul cu elementele:

a. **(1,2,3,4,5,6,9,7,10)**
b. **(1,2,3,4,5,6,7,9,10)**
c. nu se poate realiza interclasarea
d. **(1,3,5,9,10,2,4,6,7)**

2. Într-o matrice pătratică de dimensiune **n**, notăm cu **a(i,j)** elementul situat pe linia **i** şi coloana **j** (**1≤i≤n şi 1≤j≤n**). Diagonala principală şi cea secundară determină în matrice patru zone triunghiulare. Ce condiţie trebuie să îndeplinească indicii elementelor din triunghiul superior al matricei?



a. **i<j şi (i+j)<n+1**
b. **i<j sau (i+j)<n**
c. **i>j şi (i+j)<n**
d. **i+j=n şi i>j**

3. Într-o matrice cu **10** linii şi **20** de coloane, dorim să inserăm o nouă coloană având toate elemente **0** după cea de-a treia coloană a acestei matrice. Pentru a realiza această operaţie:

a. vom deplasa toate coloanele, începând de la a patra până la ultima, cu o poziţie spre dreapta şi pe coloana a patra, pe toate liniile vom memora **0**.
b. vom deplasa toate coloanele, începând de la a treia până la prima, cu o poziţie spre stânga şi pe coloana a patra, pe toate liniile vom memora **0**.
c. vom deplasa toate coloanele, începând de la ultima până la a patra, cu o poziţie spre dreapta şi pe coloana a patra, pe toate liniile vom memora **0**.
d. vom deplasa toate coloanele, începând de la ultima până la a treia, cu o poziţie spre dreapta şi pe coloana a treia, pe toate liniile vom memora **0**.

4. Linia a doua şi a cincea a unui tablou bidimensional **a** cu 10 linii şi 10 coloane **NU** coincid dacă:

a. Există **i∈{1,2,…,10}** astfel încât **a[2,i]<>a[5,i]**
b. Oricare **i∈{1,2,…,10}** astfel încât **a[i,5]=a[i,2]**
c. Oricare **i∈{1,2,…,10}** astfel încât **a[2,i]=a[5,i]**
d. Există **i∈{1,2,…,10}** astfel încât **a[i,2]<>a[i,5]**

5. Pentru interschimbarea conţinutului a două coloane **i** şi **j** ale unei matrice **a**, se copiază elementele coloanei **i** ale matricei **a** în componentele corespunzătoare ale unui vector **aux**, apoi se copiază elementele coloanei **j** peste elementele corespunzătoare ale coloanei **i** şi în final se copiază componentele vectorului **aux** peste elementele corespunzătoare ale coloanei **j**. Din punctul de vedere al gestionării memoriei, aceasta este o metodă:

a. corectă şi eficientă
b. incorectă
c. corectă, dar ineficientă
d. a cărei eficienţă depinde de valorile elementelor matricei

6. Pentru fiecare dintre cei 15 elevi ai unei clase trebuie memorate simultan într-un program mediile semestriale la cele 18 discipine studiate şi media generală semestrială a fiecăruia. O variabilă care corespunde acestei cerinţe se poate declara astfel:

a.
```
var a:array[1..15,1..19] of float;
```
b.
```
var a:array{1..270] of real;
```
c.
```
var a:array[1..15,1..18] of byte;
```
d.
```
var a:array[1..15,1..19] of real;
```

a.
```
real a[15][19];
```
b.
```
float a[270];
```
c.
```
int a[15][18];
```
d.
```
float a[15][19];
```

7. Cum trebuie declarat tabloul unidimensional **x** dacă el trebuie să reţină simultan toate valorile memorate într-o matrice care poate avea maximum **10** linii şi **10** coloane? Numerele din matrice sunt valori întregi având cel mult **3** cifre fiecare.

a.
```
Var x : array[0..99] of integer;
```
b.
```
Var x : array[1..99] of integer;
```
c.
```
Var x : array[0..19] of integer;
```
d.
```
Var x : array[0..9] of integer;
```

a.
```
int x[100];
```
b.
```
int x[99];
```
c.
```
int x[20];
```
d.
```
int x[10];
```

8. Indicii elementelor situate pe diagonala principală a unei matrice pătratice au următoarea proprietate:

a. diferenţa lor este constantă
b. suma lor este constantă
c. sunt diferiţi
d. au parităţi diferite

9. Care dintre următoarele variante reprezintă o declarare corectă a unui tablou unidimensional având identificatorul **mat** şi **100** de componente de tip caracter?

a.
```
var sir[1..100] of char;
```
b.
```
var mat: array [1..100] of byte;
```
c.
```
var mat: array [1..100] of char;
```
d.
```
var sir= array [1..100] of char;
```

a.
```
mat[100] of char;
```
b.
```
char mat[1..100];
```
c.
```
char mat[100];
```
d.
```
char[100];
```

10. Care dintre următoarele secvenţe calculează suma elementelor de pe linia **k** a unei matrice **a**, cu **m** linii (numerotate de la **1** la **m**) şi **n** coloane (numerotate de la **1** la **n**)?

a.
```
s:=0;
for i:=m downto 1 do
    s:=s+a[k,i]
```

b.
```
s:=0; i:=1;
while i<=m do begin
   s:=s+a[i,k];i:=i+1 end
```

c.
```
s:=0;
for i:=n downto 1 do
   s:=s+a[i,k]
```

d.
```
s:=0; i:=1;
while i<=n do begin
   s:=s+a[k,i];i:=i+1 end
```

a.
```
s=0;
for(i=m;i>0;i--)s=s+a[k][i];
```

b.
```
s=0; i=1;
while (i<=m)
   {s=s+a[i][k];i++;}
```

c.
```
s=0;
for(i=n;i>=1;i--)
         s=s+a[i][k];
```

d.
```
s=0; i=1;
while(i<=n)
    {s=s+a[k][i];i++;}
```

11. În urma executării secvenţei alăturate de program, se va construi matricea

```
for j:=1 to 5 do
 for i:=1 to 3 do
   a[i,j]:=i+j;
```

```
for(j=1;j<=5;j++)
 for(i=1;i<=3;i++)
   a[i][j]=i+j;
```

a.
```
2 3 4
3 4 5
4 5 6
5 6 7
6 7 8
```

b.
```
2 4 5
3 4 5
4 5 6
```

c.
```
2 3 4 5 6
3 4 5 6 7
4 5 6 7 8
```

d.
```
2 3 4 5 6
3 2 3 4 5
4 3 4 5 6
```

12. Se consideră o matrice patratică, **a**, având liniile şi coloanele numerotate de la 1 la 4.Care va fi matricea **a** în urma executării secvenţei următoare?

```
for i:=1 to 3 do
  for j:=i+1 to 4 do
     begin
       a[i,j]:=2;
       a[j,i]:=-2;
     end;
for i:=1 to 4 do a[i,i]:=1;
```

```
for(i=1;i<=3;i++)
for(j=i+1;j<=4;j++)
{
  a[i][j]=2;
  a[j][i]=-2;
}
for(i=1;i<=4;i++) a[i][i]=1;
```

a.
```
 1  2  2  2
-2  1  2  2
-2  2  1  2
-2 -2  2  1
```

b.
```
 1  2  2  2
-2  1  2  2
-2 -2  1  2
-2 -2 -2  1
```

c.
```
1  2  2  2
2  1  2  2
2  2  1  2
2  2  2  1
```

d.
```
1 -2  -2  -2
2  1  -2  -2
2  2   1  -2
2  2   2   1
```

13. Pentru o matrice **a** cu **10** linii şi **10** coloane numerotate de la **1** la **10**, stabiliţi ce calculează secvenţa de program alăturată în variabila **s**:

```
s:=0;
for i:=1 to 10 do
   for j:=i to 10 do
        s:=s+a[i,j];
```

```
s=0;
for(i=0;i<10;i++)
   for(j=i;j<10;j++)
        s=s+a[i][j];
```

a. suma elementelor situate strict deasupra diagonalei principale
b. suma elementelor situate strict deasupra diagonalei secundare
c. suma elementelor situate deasupra diagonalei principale, inclusiv diagonala principală.
d. suma elementelor situate strict sub diagonala principală

14. Pentru o matrice **a** cu **10** linii şi **10** coloane, stabiliţi ce calculează secvenţa de program alăturată în variabila **s**:

```
s:=0;
for i:=1 to 10 do
   for j:=1 to 10-i do
        s:=s+a[i,j]
```

```
s=0;
for(i=0;i<10;i++)
   for(j=0;j<9-i;j++)
s=s+a[i][j];
```

a suma elementelor situate strict sub diagonala secundară
b suma elementelor situate strict deasupra diagonalei secundare
c suma elementelor situate strict deasupra diagonalei principale
d suma elementelor situate strict sub diagonala principală

15. Dacă variabila **a** este o matrice pătratică de **n** linii şi **n** coloane cu elemente numere reale, atunci secvenţa de instrucţiuni alăturată, calculează în variabila reală **s**, suma:

```
s := 0;
for i:= 1 to n do
      s := s + a[i,i];
```

```
s = 0;
for(i = 1; i <= n; i++)
      s = s + a[i][i];
```

a. elementelor matricei
b. elementelor de pe diagonala principală
c. elementelor de sub diagonala principală
d. elementelor de pe diagonala secundară

16. Care dintre următoarele secvenţe este corectă şi declară o matrice de **400** de caractere?

a. `var a:array[1..399] of caracter;`
b. `var a of char:array[1..20, 1..20];`
c. `var a:array[1..400] of integer;`
d. `var a:array[1..20, 1..20] of char;`

a. `char a[399];`
b. `a[20][20] char;`
c. `int a[400];`
d. `char a[20][20];`

17. Fie **a** matricea pătratică cu **n** linii şi **n** coloane cu elemente numere întregi. Cu ce instrucţiune trebuie înlocuite punctele de suspensie din secvenţa de program alăturată, astfel încât aceasta să afişeze suma elementelor de pe linia **x**?

```
s := 0;
for i := 1 to n do
      ............
write(s);
```

```
s = 0;
for(i = 0; i < n; i++)
      ............
printf("%d",s);|cout<<s;
```

a. `s := s + a[i,i];`
b. `s := s + a[x,i];`
c. `s := s + a[i,x];`
d. `s := a[x,i];`

a. `s += a[i][i];`
b. `s += a[x][i];`
c. `s += a[i][x];`
d. `s = a[x][i];`

18. Matricea **mat** are **m** linii şi **n** coloane, atât liniile cât şi coloanele sunt numerotate începând de la **1** , iar **c** este un număr natural nenul mai mic sau egal cu **n**. Care dintre următoarele secvenţe de instrucţiuni calculează în variabila **s** suma elementelor coloanei **c** a matricei **mat**?

a.
```
s:=0; for i:=1 to m do
s:=s+mat[i,c];
```
b.
```
s:=0; for i:=1 to n do
s:=s+mat[c,i];
```
c.
```
s:=0; for i:=0 to n-1 do
s:=s+mat[i,c];
```
d.
```
s:=0; for i:=1 to m do
s:=s+mat[i,j];
```

a.
```
s=0; for(i=1;i<=m;i++)
        s=s+mat[i][c];
```
b.
```
s=0; for(i=1;i<=n;i++)
        s=s+mat[c][i];
```
c.
```
s=0; for(i=0;i<n;i++)
        s=s+mat[i][c];
```
d.
```
s=0; for(i=1;i<=m;i++)
        s=s+mat[i][j];
```

19. Ştiind că tabloul **a** este declarat prin

```
const a: array[0..1,0..2] of integer
         =((1,2,3),(4,5,6));
```

```
int a[2][3]={1,2,3,4,5,6};
```

stabiliţi ce se afişează în urma executării instrucţiunii:

```
write(a[0,2]+a[1,0]*a[1,1]-a[1,2]);
```

```
printf("%d",a[0][2]+a[1][0]*
        a[1][1]-a[1][2]);
/ cout<<a[0][2]+a[1][0]*
       a[1][1]-a[1][2];
```

a. 17 **b.** 13 **c.** 0 **d.** 1

2.2.2. Probleme (programe Pascal | C / C++)

1. Scrieţi programul Pascal/C/C++ care construieşte în memorie o matrice pătratică cu **n** linii şi **n** coloane formată numai din valori **1** şi **2** astfel încât elementele de pe diagonala secundară şi cea principală să fie egale cu **1**, iar restul elementelor din matrice să fie egale cu **2**. Valoarea lui **n** (număr natural, **2<n<23**) se citeşte de la tastatură, iar matricea se va afişa pe ecran, câte o linie a matricei pe câte o linie a ecranului, cu câte un spaţiu între elementele fiecărei linii (ca în exemplu).

De exemplu, pentru **n=5** se construieşte în memorie şi se afişează matricea:

```
1 2 2 2 1
2 1 2 1 2
2 2 1 2 2
2 1 2 1 2
1 2 2 2 1
```

2. Scrieţi programul Pascal/C/C++ care construieşte în memorie o matrice pătratică cu **n** linii şi **n** coloane formată numai din valori **0,1** şi **2** astfel încât elementele de pe diagonala secundară şi cea principală să fie egale cu **0,** elementele situate între diagonalele matricei, în partea superioară şi inferioară a acesteia, să fie egale cu **1**, iar restul elementelor din matrice să fie egale cu **2**. Valoarea lui **n** (număr natural, **2<n<23**) se citeşte de la tastatură, iar matricea se va afişa pe ecran, câte o linie a matricei pe câte o linie a ecranului, cu spaţii între elementele fiecărei linii (ca în exemplu).

De exemplu, pentru **n=5** se construieşte în memorie şi se afişează matricea:

```
0 1 1 1 0
2 0 1 0 2
2 2 0 2 2
2 0 1 0 2
0 1 1 1 0
```

3. Scrieţi programul Pascal/C/C++ care construieşte în memorie o matrice pătratică cu **n** linii şi **n** coloane formată numai din valori **1** şi **2** astfel încât elementele de pe diagonala secundară şi cea principală să fie egale cu **1,** elementele situate între diagonalele matricei, în partea superioară şi inferioară a acesteia, să fie egale cu **1**, iar restul elementelor din matrice să fie egale cu **2**. Valoarea lui **n** (număr natural, **2<n<23**) se citeşte de la tastatură, iar matricea se va afişa pe ecran, câte o linie a matricei pe câte o linie a ecranului, cu spaţii între elementele fiecărei linii (ca în exemplu).

De exemplu, pentru **n=5** se construieşte în memorie şi se afişează matricea:

```
1 1 1 1 1
2 1 1 1 2
2 2 1 2 2
2 1 1 1 2
1 1 1 1 1
```

4. Scrieţi programul Pascal/C/C++ care citeşte de la tastatură un număr natural impar **n** (**2<n<23**) şi construieşte în memorie o matrice pătratică cu **n** linii şi **n** coloane formată numai din valori **1, 2** şi **3** astfel încât elementele din matrice pe coloana mediană (situată în mijlocul matricei) şi linia mediană să fie egale cu **1**, elementul situat la intersecţia liniei şi coloanei mediane să fie egal cu **2,** iar restul elementelor din matrice să fie egale cu **3**. Matricea se va afişa pe ecran, câte o linie a matricei pe câte o linie a ecranului, cu spaţii între elementele fiecărei linii (ca în exemplu).

De exemplu, pentru **n=5** se construieşte în memorie şi se afişează matricea:

```
3 3 1 3 3
3 3 1 3 3
1 1 2 1 1
3 3 1 3 3
3 3 1 3 3
```

5. Scrieţi programul Pascal/C/C++ care citeşte de la tastatură un număr natural **n** (**2<n<30**) şi construieşte în memorie o matrice pătratică cu **n** linii şi **n** coloane formată numai din valori ce aparţin mulţimii **{1,2,3,...,n}** astfel încât elementele din matrice situate pe diagonala secundară să fie egale cu **n**, elementele situate pe celelalte două "semidiagonale" paralele cu diagonala secundară şi alăturate diagonalei secundare să fie egale cu **n-1**, elementele situate pe următoarele două "semidiagonale" paralele cu diagonala secundară, de o parte şi de alta a acesteia, să fie egale cu **n-2** etc. Matricea se va afişa pe ecran, câte o linie a matricei pe câte o linie a ecranului, cu spaţii între elementele fiecărei linii (ca în exemplu).

De exemplu, pentru **n=5** se construieşte în memorie şi se afişează matricea:

```
1 2 3 4 5
2 3 4 5 4
3 4 5 4 3
4 5 4 3 2
5 4 3 2 1
```

6. Scrieţi un program care construieşte în memorie o matrice pătratică de dimensiune **n** (**1≤n≤30**) care să conţină pe prima linie, în ordine crescătoare, numerele de la **1** la **n**, pe a doua linie, în ordine descrescătoare, numerele de la **2*n** la **n+1**, pe a treia linie, în ordine crescătoare, numerele de la **2*n+1** la **3*n**, etc. Valoarea lui **n** se citeşte de la tastatură, iar matricea se va afişa pe ecran, câte o linie a matricei pe câte o linie a ecranului, cu spaţii între elementele fiecărei linii, după cum se poate observa în exemplu.

De exemplu, pentru **n=3** se construieşte în memorie şi se afişează matricea:

```
1 2 3
6 5 4
7 8 9
```

7. Scrieţi un program care citeşte un număr natural **n** (**1≤n≤10**) şi apoi construieşte în memorie o matrice cu **n** linii şi **n** coloane care va conţine pe fiecare linie numerele de la **1** la **n**, dispuse astfel încât pe orice linie elementele să fie distincte şi pe orice coloană elementele să fie distincte. Afişaţi matricea pe ecran, câte o linie a matricei pe câte o linie a ecranului, cu câte un spaţiu între elementele fiecărei linii (ca în exemplul alăturat).

Exemplu. Pentru **n=4** o soluţie posibilă este:

```
1 2 3 4
2 3 4 1
4 1 2 3
3 4 1 2
```

8. Scrieţi un program care citeşte un număr natural nenul **n** (**n<11**) şi apoi construieşte în memorie o matrice cu **n** linii şi **n** coloane astfel:

- pe diagonala principală toate elementele au valoarea **0**;
- sub diagonala principală, respectiv deasupra diagonalei principale elementele vor conţine valorile **1, 2, 3, …,** $\frac{n\cdot(n-1)}{2}$, dispuse în ordine crescătoare pe linii, iar în cadrul unei linii, în ordine crescătoare de la stânga la dreapta, ca în exemplu. Programul va afişa matricea pe ecran, câte o linie a matricei pe o linie a ecranului, cu un spaţiu între elementele fiecărei linii.

Exemplu: pentru **n=5,** programul va afişa matricea următoare:

```
0 1 2 3 4
1 0 5 6 7
2 3 0 8 9
4 5 6 0 10
7 8 9 10 0
```

9. Scrieţi un program care citeşte de la tastatură un număr natural **n** (**n≤20**) şi construieşte în memorie o matrice pătratică având **n** linii şi **n** coloane, cu elemente **0** şi **1**, dispuse în pătrate concentrice, fiecare pătrat fiind format doar din valori **1** sau doar din valori **0**, ca în exemplul alăturat, astfel încât elementul aflat pe prima linie şi prima coloană să fie egal cu **1**. Afişaţi matricea pe ecran câte o linie a matricei pe o linie a ecranului, cu un spaţiu între elementele fiecărei linii (ca în exemplul alăturat).

Exemplu. Pentru **n=6** o soluţie posibilă este:

```
1 1 1 1 1 1
1 0 0 0 0 1
1 0 1 1 0 1
1 0 1 1 0 1
1 0 0 0 0 1
1 1 1 1 1 1
```

10. Scrieţi programul Pascal/C/C++ care citeşte de la tastatură un număr natural **n** (**2<n<30**) şi construieşte în memorie o matrice pătratică cu **n** linii şi **n** coloane ale cărei elemente vor primi valori după cum urmează:

- elementele aflate pe diagonala secundară a matricei vor primi valoarea **n+1**;
- elementele de pe prima linie, cu excepţia celui aflat pe diagonala secundară vor primi valoarea **1**;
- elementele de pe a doua linie, cu excepţia celui aflat pe diagonala secundară vor primi valoarea **2**, etc.

Programul va afişa matricea astfel construită pe ecran, câte o linie a matricei pe câte o linie a ecranului, cu spaţii între elementele fiecărei linii (ca în exemplu).

De exemplu pentru **n=4** matricea va conţine:

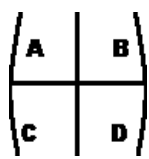
```
1  1  1  5
2  2  5  2
3  5  3  3
5  4  4  4
```

11. Scrieţi un program Pascal/C/C++ care să citească de la tastatură două numere naturale **n** si **m** (**0<n<=5; 0<m<=5**), şi care construieşte în memorie o matrice cu **n** linii, numerotate de la **1** la **n** si **m** coloane numerotate de la **1** şi **m** ce conţine toate numerele naturale de la **1** la **n*m**, astfel încât parcurgând matricea pe linii, de la prima la ultima şi fiecare linie numerotată cu un număr de ordine impar parcurgându-se de la stânga la dreapta iar fiecare linie numerotată cu un număr de ordine par parcurgându-se de la dreapta la stânga, se obţine şirul tuturor numerelor naturale consecutive de la **1** la **n*m** (ca în exemplu). Matricea se va afişa pe ecran, câte o linie a matricei pe câte o linie a ecranului, cu spaţii între elementele fiecărei linii (ca în exemplu).

Exemplu. Pentru **n=4, m=3** matricea construită este:

```
1   2   3
6   5   4
7   8   9
12  11  10
```

12. O matrice pătrată, cu **2*n-1** linii şi **2*n-1** coloane, este împărţită în patru zone notate **A, B, C, D,** de linia **n** şi coloana **n**, conform figurii alăturate. Elementele liniei **n** şi coloanei **n** nu aparţin nici uneia dintre zone. Scrieţi programul



Pascal/C/C++ care citeşte de la tastatură două numere naturale **n** şi **m** **(1<n≤10,1000≤m<10000)**, formează în memorie şi apoi afişează pe ecran o matrice pătrată cu **2*n-1** linii şi **2*n-1** coloane, în care elementele de pe linia **n** şi coloana **n** sunt egale cu **0**, elementele care aparţin zonei **A** sunt egale cu cifra unităţilor numărului **m**, elementele care aparţin zonei **B** sunt egale cu cifra zecilor numărului **m**, elementele care aparţin zonei **C** sunt egale cu cifra sutelor numărului **m** şi elementele care aparţin zonei **D** sunt egale cu cifra miilor numărului **m**.

De exemplu pentru **n=3** şi **m=3681** se va afişa matricea din caseta alăturată:

```
1 1 0 8 8
1 1 0 8 8
0 0 0 0 0
6 6 0 3 3
6 6 0 3 3
```

13. Se citesc de la tastatură două valori naturale **m** şi **n** (**m,n<=100**) şi apoi **m*n** numere întregi mai mici de **32000** reprezentând elementele unei matrice cu **m** linii şi **n** coloane. Se cere să se determine valorile maxime de pe fiecare linie a matricei şi să se scrie pe ecran suma lor.

De exemplu, pentru **m=3**, **n=5** şi matricea $\begin{pmatrix} 1 & 9 & 3 & 4 & 7 \\ 9 & 6 & 12 & 5 & 10 \\ 5 & 2 & 7 & 1 & 3 \end{pmatrix}$, se afişează pe ecran **28** (deoarece valorile maxime pe linii sunt **9,12,7,** iar suma lor este **28**)

14. Scrieţi un program **Pascal** care construieşte în memorie o matrice pătrată cu **n** linii şi **n** coloane formată astfel:
- elementele aflate pe diagonala principală sunt toate nule.
- elementele de pe linia **i** (**1<=i<=n**), aflate deasupra diagonalei principale au valoarea egală cu **i**;
- elementele de pe coloana **i** (**1<=i<=n**), aflate sub diagonala principală au valoarea egală cu **i**.

Valoarea lui **n** (număr natural, **2<n<50**) se citeşte de la tastatură, iar matricea construită se afişează pe ecran, câte o linie a matricei pe o linie a ecranului. Între două elemente ale fiecărei linii se va lăsa un spaţiu. De exemplu, pentru **n=4** se va afişa:

```
0 1 1 1
1 0 2 2
1 2 0 3
1 2 3 0
```

15. Se citesc de la tastatură două numere naturale **n** şi **m** (**1<m<10, 1<n<10**) şi o matrice **a** cu **n** linii şi **m** coloane formată din numere întregi de cel mult **4** cifre fiecare. Scrieţi programul **Pascal/C/C++** ce sortează descrescător elementele fiecărei linii. Matricea sortată se va afişa pe ecran, fiecare linie a matricei pe câte o linie a ecranului, elementele unei linii fiind separate prin spaţii.

Exemplu: Pentru **n=3** şi **m=5** matricea:

```
 6  2  9  1  2                      9  6  2  2  1
-3  0  1  4  4        se va afişa   4  4  1  0 -3
 9  1  3  2  7                      9  7  3  2  1
```

16. Se citesc de la tastatură un număr natural **n** (**1<n<10**) şi o matrice pătratică cu **n** linii şi **n** coloane formată din numere întregi de maximum **4** cifre. Scrieţi programul **Pascal/C/C++** ce sortează descrescător doar elementele situate pe diagonala principală. Matricea sortată se va afişa pe ecran, fiecare linie a matricei pe câte o linie a ecranului, elementele unei linii fiind separate prin câte un spaţiu.

Exemplu: Pentru **n=3** şi o matrice formată din elementele:

```
2 3 1                          8 3 1
5 8 2     programul va afişa:  5 3 2
6 2 3                          6 2 2
```

17. Scrieţi un program Pascal/C/C++ care citeşte de la tastatură un număr natural **n** (**n<=10**) şi elementele (numere naturale mai mici decât **100**) ale unui tablou bidimensional cu **n** linii şi **n** coloane; programul va determina şi va afişa pe ecran câte numere pare sunt situate pe „rama" tabloului. Rama unei matrice este formată din prima linie, prima coloană, ultima linie şi ultima coloană a acesteia.

Exemplu: pentru **n=4** şi tabloul:

```
 1  2  3  4                    se va afişa: 6
 5  6  7  8
 9 10 11 12
13 14 15 16
```

18. O matrice binară este o matrice în care orice element are valoarea **0** sau **1**. Pentru o valoarea naturală **n** (**3 ≤ n ≤ 20**) citită de la tastatură, se doreşte alcătuirea unei matrice binare cu **n** coloane şi **2n-2** linii cu următoarele proprietăţi:

- nu există două linii identice în matrice;
- două linii consecutive din matrice au **n-2** poziţii cu valori identice iar celelalte **2** poziţii conţin valori diferite.

a) Faceţi o descriere în limbaj natural pentru un algoritm care construieşte în memorie o matrice binară care îndeplineşte cerinţele anterioare. Explicaţi strategia prin care construiţi matricea, justificând că aceasta respectă condiţiile din enunţ.

b) Scrieţi un program care implementează algoritmul descris la punctul a). Matricea construită se va afişa pe ecran, câte o linie a matricei pe câte o linie a ecranului, cu un spaţiu între elementele fiecărei linii (ca în exemplu).
Exemplu: dacă se citeşte pentru **n** valoarea **3**, o matrice care îndeplineşte condiţiile din enunţ este:

```
0 0 1
1 0 0
0 1 0
1 1 1
```

19. Se consideră un tablou bidimensional **A** cu **200** de linii şi **200** de coloane (cu liniile şi coloanele numerotate de la **1** la **200**) în care pe fiecare linie toate elementele au valoarea **0**, cu excepţia celor de pe diagonala principală şi a celor aflate imediat în stânga şi imediat în dreapta acestora. Elementele nenule sunt completate în ordine, linie cu linie de sus în jos şi pe aceeaşi linie de la stânga la dreapta, cu valorile naturale **1, 2, 3 ...** ca în exemplu:

1	2	0	0	0	...	0	0	0
3	4	5	0	0	...	0	0	0
0	6	7	8	0	...	0	0	0
...								
0	0	0	0	0	...	594	595	596
0	0	0	0	0	...	0	597	598

Pentru tabloul descris anterior, se citeşte un număr natural **k (1≤k≤598)**. Se cere să se afişeze două numere naturale reprezentând linia şi respectiv coloana pe care se află valoarea **k** în **acest tablou.**

a) Descrieţi în limbaj natural un algoritm eficient de rezolvare (din punct de vedere al memoriei utilizate şi al timpului de executare) şi explicaţi în ce constă eficienţa lui (în **3-4** rânduri).

b) Scrieţi programul corespunzător algoritmului ales.

Exemplu: pentru **k=5** se afişează: **2 3** (deoarece valoarea **5** se află pe linia **2** şi coloana **3** a tabloului).

20. În fişierul **MATRICE.IN** se află memorate elementele unei matrice; pe prima linie se află numerele naturale **n** şi **m** (**n≤10, m≤10**) separate printr-un spaţiu, care reprezintă numărul de linii, respectiv coloane ale matricei, iar pe următoarele **n** linii câte **m** numere întregi, separate prin câte un spaţiu, reprezentând elementele matricei. Scrieţi programul **PASCAL/C/C++** care citeşte din fişierul **MATRICE.IN** elementele matricei şi afişează pe ecran numărul coloanelor tabloului formate numai din valori distincte.**)**

Exemplu:

```
MATRICE.IN
4 5
1 2 1 5 10
7 3 3 2  8
6 3 4 9 15
4 7 1 12 0
```

Trei coloane au elemente distincte (**1,4** şi **5**). Pe ecran se va afişa mesajul: **3**

21. Se citeşte un număr natural **n** (**0<n<11**) şi apoi un tablou unidimensional **v** cu **n*(n-1)/2** elemente numere întregi. Scrieţi un program care să construiască un tablou bidimensional **a** cu **n** linii şi **n** coloane, simetric faţă de diagonala principală, astfel încât prin parcurgerea zonei aflate strict deasupra diagonalei principale, linie cu linie, să se obţină, în ordine, elementele tabloului **v**. Pe diagonala principală a tabloului **a** toate elementele sunt **0**. Să se afişeze pe ecran tabloul **a** construit.

De exemplu, dacă **n=4 şi v=(1,2,3,4,5,6)**, atunci **a** este:

```
0 1 2 3
1 0 4 5
2 4 0 6
3 5 6 0
```

22. Un tablou bidimensional **a** cu **m** linii (**0<m<11**) şi **n** coloane (**0<n<21**) se numeşte **palindromic** dacă, şirul format prin parcurgerea sa linie cu linie, are primul element al parcurgerii egal cu ultimul element al parcurgerii, al doilea egal cu penultimul, etc. Să se scrie un program **PASCAL/C/C++** care citeşte două numere **m** şi **n** şi apoi elementele tabloului bidimensional **a** de la tastatură şi afişează pe ecran mesajul **„DA”** în cazul în care tabloul **a** este palindromic şi **„NU”** în caz contrar.

De exemplu dacă tabloul **a** citit este cel alăturat, se va afişa mesajul **„DA”.**

```
1 2 3
4 5 4
3 2 1
```

23. Scrieţi un program **PASCAL/C/C++** care citeşte un tablou bidimensional cu **n** linii (**0<n<10**) şi **m** coloane (**0<m<20**) şi elemente numere întregi, un număr natural **k** şi afişează cel mai mare element de pe linia **k**, numerotarea liniilor începând de sus în jos, de la **1** la **n**.

24. Se citesc **2** numere naturale nenule **m,n (2<m,n<10).** Să se scrie programul **PASCAL/C/C++** care construieşte în memorie o matrice **A** cu **m** linii şi **n** coloane ce conţine primele **m*n** numere naturale impare. Prima linie a matricei va conţine, în ordine strict crescătoare, primele **n** numere impare, a doua linie va conţine, în ordine strict crescătoare, următoarele **n** numere impare, etc. Matricea se va afişa pe ecran, câte o linie a matricei pe câte o linie a ecranului, elementele fiecărei linii fiind separate prin spaţii.
De exemplu pentru **m=3** şi **n=4** se va afişa matricea următoare:

```
 1  3  5  7
 9 11 13 15
17 19 21 23
```

25. Se citesc 2 numere naturale nenule **m,n (2<m,n<10).** Să se scrie programul **PASCAL/C/C++** care construieşte în memorie o matrice A cu **m** linii şi **n** coloane cu proprietatea că **A[i][j]** este cel mai mic număr care se poate obţine prin concatenarea lui **i** cu **j**..Matricea se va afişa pe ecran, câte o linie a matricei pe câte o linie a ecranului, elementele fiecărei linii fiind separate prin spaţii.
De exemplu pentru **m=3** şi **n=4** se va afişa matricea următoare:

```
11 12 13 14
12 22 23 24
13 23 13 34
```

26. Scrieţi programul **Pascal/C/C++** care construieşte în memorie o matrice pătrată cu **n** linii şi **n** coloane formată numai din valori cuprinse între **1** şi **n** astfel încât să nu existe două elemente egale pe aceeaşi linie şi nici pe aceeaşi coloană.
Prima linie trebuie să conţină, în această ordine, numerele **1,2,…,n**; linia a doua va avea, în ordine, numerele **2,3,…,n,1**; linia a treia va avea, în ordine, numerele **3,4,…,n,1,2,** iar pe ultima linie numerele **n,1,2,3,…,n-1**.Valoarea lui **n** (număr natural, **2<n<25**) se citeşte de la tastatură, iar matricea construită se afişează, pe linii, pe ecran.

De exemplu, pentru **n=4** se va afişa:

```
1 2 3 4
2 3 4 1
3 4 1 2
4 1 2 3
```

27. Pentru o valoare **n** (număr natural, **1<n<20**) citită de la tastatură se cere să se scrie un program **Pascal/C/C++** care construieşte în memorie o matrice cu **n** linii şi **n** coloane ale cărei elemente sunt numerele de la **1** la $\mathbf{n}^2$, aşezate în ordine crescătoare, pe coloane, începând cu prima coloană. Elementele matricei se afişează pe ecran, pe linii, ca în exemplu.

De exemplu, pentru **n=3**, se va construi şi afişa matricea:

```
1 4 7
2 5 8
3 6 9
```

28. Scrieţi un program **Pascal/C/C++** care citeşte de la tastatură un număr natural **n (2<n<21)** şi apoi **n** linii cu câte **n** numere întregi de cel mult **7** cifre ce formează un tablou bidimensional **a**.
Să se afişeze pe ecran diferenţa dintre suma elementelor de pe diagonala principală şi suma elementelor de pe diagonala secundară a matricei **a**.

29. Se citeşte de la tastatură un număr natural **n** (**0 < n < 100**) apoi **n*n** valori din mulţimea **{1,0}**, care se memorează într-o matrice cu **n** linii şi **n** coloane. Să se afişeze pe ecran numărul de ordine al liniei, care conţine cel mai mare număr de cifre de **1**. Dacă două sau mai multe linii conţin acelaşi număr de cifre de **1**, se va afişa numai numărul de ordine al primei dintre aceste linii.

Pentru **n = 4** şi matricea

```
0 1 0 0
1 0 1 1
0 1 0 1
0 1 1 1
```

se va afişa **2**

30. Scrieţi un program **Pascal/C/C++** care citeşte de la tastatură un număr natural **n (2<=n<=100)** şi apoi **n** linii cu câte **n** numere naturale de cel mult **5** cifre ce formează un tablou bidimensional **a**. Să se afişeze pe ecran, separate printr-un spaţiu, elementele tabloului **a** care au proprietatea că divid suma vecinilor existenţi (pe linie, coloană şi diagonale). Elementele vor fi identificate prin parcurgerea tabloului **a** pe linii, de sus in jos, pe fiecare linie parcurgerea făcându-se de la stânga la dreapta.

Ex.: Pentru valorile **n=3** şi tabloul **a:**

```
2 5 4
1 2 8
2 4 2
```

se va afişa: **2 1 2 2**

31. Scrieţi programul Pascal/C/C++ care citeste de la tastatură un număr natural **n (2<n<25)** şi construieşte în memorie o matrice pătratică cu **n** linii şi **n** coloane formată numai din valori **0,1** şi **2** astfel încât elementele de deasupra

diagonalei principale să fie egale cu **0**, cele de pe diagonala principală să fie egale cu **1**, iar cele de sub aceasta să fie egale cu **2**. Matricea se va afişa pe ecran, fiecare linie a matricei pe o linie a ecranului şi valorile de pe aceeaşi linie separate printr-un singur spaţiu ca în exemplul de mai jos.

Pentru **n=5** se construieşte în memorie şi se afişează matricea:

```
1 0 0 0 0
2 1 0 0 0
2 2 1 0 0
2 2 2 1 0
2 2 2 2 1
```

32. Scrieţi programul C/C++ care construieşte în memorie o matrice pătratică cu **n** linii şi **n** coloane ale cărei componente sunt numere naturale nenule cuprinse între **1** şi **n**, astfel încât elementele fiecărei linii sunt distincte două câte două şi de asemenea pe orice coloană nu există două elemente egale. Valoarea lui **n** (număr natural, **1<n<100**) se citeşte de la tastatură, iar matricea se va afişa pe ecran, pe linii.

De exemplu, pentru **n=2** se poate construi şi afişa matricea:

```
1 2
2 1
```

33. Scrieţi un program Pascal/C/C++ care citeşte de la tastatură două numere naturale **m** şi **n** (**1<m<10** şi **1<n<10**) şi construieşte în memorie, apoi afişează pe ecran o matrice având **m** linii şi **n** coloane, ale cărei componente sunt obţinute prin concatenarea cifrelor reprezentând linia şi respectiv coloana pe care se află (de exemplu, elementul aflat pe linia **3** şi coloana **7** va fi **37**). Atât liniile matricei, cât şi coloanele se numerotează începând de la **1**, iar matricea se va afişa pe ecran câte o linie a matricei pe câte o linie a ecranului, cu spaţii între elementele fiecărei linii (ca în exemplu).
De exemplu, pentru **m=2,n=3** se va construi şi afişa matricea:

```
11 12 13
21 22 23
```

2.3. Tipul şir de caractere

2.3.1. Teste grilă (Pascal | C / C++)

1. Care dintre următoarele secvenţe reprezintă o declarare de variabilă aparţinând unui tip de date structurat?

a. `x:real;`	**b.** `x:integer;`	**a.** `float x;`	**b.** `int x;`
c. `x:char;`	**d.** `x:string;`	**c.** `char x;`	**d.** `char x[3];`

2. Care dintre următoarele secvenţe reprezintă declararea corectă şi eficientă a unei variabile **x** ce reţine simultan iniţialele unui elev care este identificat cu ajutorul unui şir format din cel mult 50 de caractere, şir ce include un nume de familie şi cel mult două prenume.

a. `x:string;`	**a.** `string x;`
b. `x:string[50];`	**b.** `char x[51];`
c. `x:string[3];`	**c.** `char x[4];`
d. `x:char;`	**d.** `char x;`

3. Cum trebuie declarată variabila **x** astfel încât ea să poată reţine simultan primele zece litere mari ale alfabetului englez?

a. `Var x : string[10];`	**a.** `char x[11];`
b. `Var x : char;`	**b.** `char x;`
c. `Var x : char[10];`	**c.** `char x['A'..'J'];`
d. `Var x : array[1..10] of word;`	**d.** `float x[10];`

4. Pentru a iniţializa variabila **n** cu lungimea efectivă a şirului de caractere memorat de variabila **w** scriem instrucţiunea:

a. `n:=strlen(w)`	**a.** `n=length(w);`
b. `length(w):=n`	**b.** `strlen(w)=n;`
c. `n=ord(w[0])`	**c.** `n=(int)w[0];`
d. `n:=length(w)`	**d.** `n=strlen(w);`

5. Ştiind că variabila **s** va fi folosită pentru a memora simultan numele celor **10** profesori ai unei clase şi că fiecare dintre aceste nume are cel mult **20** de litere, care este varianta corectă de declarare a lui **s**?

a. `var s:array[1..10] of string[10];`	**a.** `char s[20][11];`
b. `var s:string[10];`	**b.** `char *s[10][21];`
c. `var s:array[1..10] of string[20];`	**c.** `char s[10][21];`
d. `var s:string[10][20];`	**d.** `STRING s[10];`

6. Pentru a compara lungimile efective a două şiruri de caractere se utilizează subprogramul predefinit:

a.	`copy`	**b.**	`delete`	**a.**	`strcmp`	**b.**	`strtok`
c.	`length`	**d.**	`insert`	**c.**	`strlen`	**d.**	`strcat`

7. Care din următoarele subprograme predefinite realizează concatenarea a două şiruri de caractere?

a.	`concat`	**b.**	`pos`	**a.**	`strcat`	**b.**	`strcmp`
c.	`strcat`	**d.**	`paste`	**c.**	`concat`	**d.**	`paste`

8. Pentru a cauta un şir de caractere în alt şir de caractere se utilizează subprogramul predefinit:

a.	`insert`	**b.**	`copy`	**a.**	`strcat`	**b.**	`strchr`
c.	`pos`	**d.**	`in`	**c.**	`strstr`	**d.**	`strcmp`

9. Două şiruri de caractere `s` şi `t` au lungimile egale dacă şi numai dacă:

a.	`length(s)=length(t)`	**a.**	`strlen(s)==strlen(t)`
b.	`^s=^t`	**b.**	`s==t`
c.	`s[1]=t[1]`	**c.**	`s[0]==t[0]`
d.	`s=t`	**d.**	`strcmp(s,t)==0`

10. Dacă `s` este o variabilă de tip şir de caractere (declarată astfel: `var s:string[100];/ char s[100];`), atunci care dintre următoarele expresii reprezintă ultimul caracter memorat în şir?

a.	`s[length(s)-1]`	**a.**	`s[strlen(s)]`
b.	`s[length(s)]`	**b.**	`s[strlen(s)-1]`
c.	`s[0]`	**c.**	`s[100]`
d.	`s[length[s]]`	**d.**	`s[strlen[s]]`

11. În secvenţa de instrucţiuni următoare, variabila `s` memorează un şir de caractere, iar variabila `i` este de tip `integer/int`. Ştiind că în urma executării secvenţei s-a afişat succesiunea de caractere `p*r*o*b*a***e*` stabiliţi care este şirul de caractere memorat de variabila `s`.

```
for i:=1 to length(s) do
        write(s[i],'*')
```

```
for(i=0;i<strlen(s);i++)
  printf("%c*",s[i]);
       /  cout<<s[i]<<'*';
```

a. `proba*e`
b. `*p*r*o*b*a***e`
c. `probae`
d. `p*r*o*b*a*e*`

12. Variabila `s` a fost declarată astfel:

```
s:string[20];
```

```
char s[20];
```

Ce se afişează după executarea secvenţei de mai jos?

```
s:='bacalaureat';
write(s[length(s)-3]);
```

```
strcpy(s,"bacalaureat");
cout<<s[strlen(s)-4];
   /  printf("%c",s[strlen(s)-4]);
```

a. r **b.** e **c.** 17 **d.** 8

13. Variabila **s** a fost declarată astfel:

```
s:string[10];
```

```
char s[10];
```

iar variabila **s** reţine şirul de caractere **bac2007**. Ce valoare are expresia

```
length(s)
```

```
strlen(s)
```

a. 8 **b.** 10 **c.** 9 **d.** 7

14. Ce se va afişa după executarea secvenţei de program de mai jos?

Varianta Pascal

```
var b : string[20];
begin
 b:='informatica'; delete(b,pos('r',b),pos('a',b));
 write(b)
end.
```

Varianta C/C++

```
char a[20]="informatica", b[20]="";
strncat(b,a,strlen(strchr(a,'t')));
cout<<b;  / printf("%s",b);
```

a. tica **b.** form **c.** rmatica **d.** info

15. Care dintre următoarele expresii este **adevărată/negativă**?

a. `length('casa')<pos('casa','acasa')`

b. `pos('a','casa')=pos('a','acasa')`

c. `'casa'>'acasa'`

d. `'casa' = 'CASA'`

a. `strlen("casa")`

b. `strcmp("casa","acasa")`

c. `strcmp("acasa","casa")`

d. `strlen("casa")-strlen("CASA")`

16. Variabilele **a** şi **b** sunt declarate astfel:

```
a,b:string[20];
```

```
char a[20],b[20];
```

Ce se afişează în urma executării secvenţei de mai jos?

```
a:='bac20'; b:='07';
write(concat(a,'*'),b);
```

```
strcpy(a,"bac20"); strcpy(b,"07");
cout<<strcat(a,"*")<<b;
|printf("%s%s",strcat(a,"*"),b);
```

a. bac20*07 **b.** bac2007 **c.** bac20bac2007 **d.** bac140

17. Pentru declararea unei variabile care memorează **20** de cuvinte având maximum **10** caractere fiecare, vom utiliza sintaxa:

a.	`var c[10,20]:string;`	**a.**	`char c[10][20];`
b.	`var c:array[1..10] of string;`	**b.**	`char c[10][ ];`
c.	`var c[20,10]:string;`	**c.**	`char c[20,10];`
d.	`var c:array[1..20] of string[10];`	**d.**	`char c[20][11];`

18. Se consideră un şir de caractere **s** de lungime maximă 20 , ce conţine cel puţin un caracter **'c'.**Care dintre următoarele secvenţe afişează poziţia primei apariţii a lui **'c'** în şirul de caractere **s**?

a.	`write(pos('c',s));`	**a.**	`cout<<strchr(s,'c')-s;` `/ printf("%d", strchr(s,'c')-s);`
b.	`write(pos(s,'c'));`	**b.**	`cout<<strrchr(s,'c');` `/ printf("%d", strrchr(s,'c'));`
c.	`write(pos(s,'c')-1);`	**c.**	`cout<<strchr(s,'c')-s-1;` `/ printf("%d", strchr(s,'c')-s-1);`
d.	`write(substr(s,'c'));`	**d.**	`cout<<strchr(s,'c');` `/ printf("%s", strchr(s,'c'));`

19. Ce se va afişa după executarea următoarei secvenţe de instrucţiuni:

```
const s:string[30]=
    'examen de bacalaureat';
begin
   s[2]:='X'; write(s);
end.
```

```
char s[30]=
"examen de bacalaureat";
s[1]='X';
cout<<s;  |  printf("%s",s);
```

a. `examen de bacalaureat`
b. `Examen de Bacalaureat`
c. `eXamen de bacalaureat`
d. `Xxamen de bacalaureat`

20. Se consideră secvenţa alăturată.Ce se va afişa dacă şirul **a** este **aabbddeff** iar şirul **b** este **aabbdeef**?

```
if (a>b) then write (a)
else
     if (a=b) then

write('egalitate')
     else write(b)
```

```
if (strcmp(a,b)>0)
   printf("%s ",a);/cout<<a;
else
  if (strcmp(a,b)==0)
      printf("egalitate");
          / cout<<"egalitate";
  else
    printf("%s ",b);/cout<<b;
```

a. `aabbdeeff`
b. `aabbddeff`
c. `nici una dintre variantele propuse`
d. `egalitate`

21. Ce se va afişa în urma executării secvenţei alăturate, dacă **x** este o variabilă de tip **string**, iar **i** are tipul **integer** ?

```
x := 'bacalaureat';
for i:= 1 to 4 do
   delete(x, 1, 1);
write(x);
```

```
char x[] = "bacalaureat";
for (int i = 1; i <= 4; i++)
      strcpy(x, x + 1);
printf("%s", x); | cout << x;
```

a. alaureat **b.** aureat **c.** laureat **d.** blaureat

22. Care dintre următoarele şiruri de caractere poate fi memorat în variabila **x** astfel încât expresia:

```
length(x)>pos('s',x)
```

```
strlen(x)>strchr(x,'s')-x+1
```

să aibă valoarea **false/0**?

a. dedus **b.** pastos **c.** sedus **d.** sarm

23. Algoritmul următor testează dacă şirul de caractere **s** este palindrom (citit atât de la stânga la dreapta, cât şi de la dreapta la stânga, se obţine acelaşi şir de caractere). Care este expresia care poate înlocui punctele de suspensie astfel încât algoritmul să fie corect ?

```
i:=1; j:=length(s);
while(i<j) and (s[i]=s[j]) do
begin i:=i+1;j:=j-1 end;
if ..... then
     write('palindrom')
 else
write('nu este palindrom');
```

```
i=0; j=strlen(s)-1;
while(i<j && s[i]==s[j])
 {i=i+1; j=j-1;}
if(.....)
    printf("palindrom");
      / cout<<"palindrom";
else
  printf("nu este palindrom");
      / cout<<"nu e palindrom";
```

a. i=j **b.** i>=j **c.** i<j **d.** i<>j

2.3.2. Probleme (Pascal | C/C++)

1. Scrieţi un program care citeşte de la tastatură un şir de caractere format din cel mult **20** de litere mici din alfabetul englez şi care afişează mesajul **"DA"** dacă şirul este format dintr-un număr egal de consoane şi vocale, respectiv mesajul **"NU"** în caz contrar.

2. Scrieţi un program care citeşte de la tastatură două şiruri de caractere, fiecare şir fiind format din cel mult **100** de litere mici din alfabetul englez, şi care afişează mesajul **"DA"** dacă toate literele din primul şir se găsesc în cel de-al doilea şir, nu neapărat în aceiaşi ordine şi de acelaşi număr de ori, sau mesajul **"NU"** în caz contrar. De exemplu, dacă primul şir este **"baraca"**, iar cel de-al doilea şir este **"abracadabra"**, programul trebuie să afişeze mesajul **"DA"** deoarece literele primului şir apar în cel de-al doilea şir.

3. Să se realizeze un program Pascal/C/C++ care citeşte de la tastatură un număr natural **n** cu cel mult nouă cifre nedivizibil cu **10** şi care afişează pe ecran numărul obţinut din **n** prin schimbarea primei cifre a numărului cu ultima cifră a sa, precum şi valoarea radicalului de ordinul **2** din numărul obţinut, cu cel mult două zecimale. Cele două numere se vor afişa pe o linie, separate printr-un spaţiu.

Exemplu: pentru **n=6094** se vor afişa: **4096 64.**

4. Scrieţi un program care citeşte de la tastatură două numere naturale **a** şi **b** (**0≤a,b≤200000000**) şi în cazul în care există cel puţin o aceeaşi cifră în ambele numere, afişează mesajul **DA**, iar în caz contrar afişează mesajul **NU**.

De exemplu, pentru **a=83** şi **b=119** se va afişa pe ecran mesajul **NU**, iar pentru **a=5003** şi **b=720** se va afişa **DA**.

5. Scrieţi programul Pascal/C/C++ care citeşte de la tastatură un text **t** format din cel mult 80 de caractere, litere şi spaţii, şi apoi elimină spaţiile multiple din textul **t**. Dacă între două cuvinte există două sau mai multe caractere spaţiu, trebuie eliminate unele dintre ele astfel încât să rămână exact unul. Se ştie că la începutul şi la sfârşitul şirului nu sunt spaţii. Textul obţinut după eliminare se va afişa pe ecran.

De exemplu, dacă se citeşte textul: **Maria□ □ □a□adus□ □cana**, atunci textul afişat va fi: **Maria□a□adus□cana**.

6. Se citesc de la tastatură două şiruri de caractere formate din cel mult **50** de litere fiecare. Să se afişeze pe ecran şirul format prin preluarea alternativă, din fiecare şir, a câte unei litere (prima literă a primului şir, apoi prima literă a celui de-al doilea, apoi a doua literă a primului şir, apoi a doua literă a celui de-al doilea şir etc). Când se epuizează literele din unul dintre şiruri, se vor prelua toate literele rămase din celălalt şir.
Dacă se citesc şirurile **ABC** şi **MNPRTXB** se va afişa şirul **AMBNCPRTXB**.

7. Pentru două şiruri de caractere cu maximum **250** de caractere fiecare (litere mici ale alfabetului englez), cu caracterele în ordine alfabetică, introduse de la tastatură, se cere să se afişeze pe ecran un al treilea şir, format din toate caracterele primelor două, aşezate în ordine alfabetică. Alegeţi un algoritm de rezolvare, eficient din punct de vedere al timpului de executare.

De exemplu pentru **sir1="ampstz"** şi **sir2="bfgostx"** se va afişa **abfgmopssttxz**.

a) Descrieţi strategia de rezolvare şi justificaţi eficienţa algoritmului ales, folosind limbajul natural (5-6 rânduri).

b) Scrieţi programul **Pascal/C/C++** corespunzător metodei descrise.

8. Scrieţi programul **Pascal/C/C++** care citeşte de la tastatură un şir de cel mult 100 de caractere, format doar din literele mici ale alfabetului englez şi caractere spaţiu. Programul afişează pe ecran şirul codificat astfel:

- după fiecare vocală se adaugă consoana imediat următoare (după **a** se inserează **b**, după **i** se inserează **j** ş.a.m.d.),
- la sfârşitul şirului se adaugă grupul de litere **stop**.

Exemplu: dacă şirul iniţial este „**azi este proba de informatica**”, după codificare se afişează: „**abzij efstef propbab def ijnfoprmabtijcabstop**”

9. Se citeşte de la tastatură un text format din cuvinte separate între ele prin câte un spaţiu. Fiecare cuvânt are cel mult 40 de caractere, doar litere mici ale alfabetului englez. Textul are cel mult 200 de caractere. Scrieţi programul **Pascal/C/C++** care afişează pe ecran, pe linii separate, doar cuvintele din textul citit care conţin cel mult trei vocale. Se consideră vocale: **a**, **e**, **i**, **o**, **u**.

Exemplu. Dacă textul este:

pentru examenul de bacalaureat se folosesc tablouri

se afişează pe ecran:

pentru
de
se
folosesc

10. Scrieţi programul **Pascal/C/C++** care citeşte de la tastatură un cuvânt de cel mult **15** litere mici ale alfabetului englez şi care afişează pe ecran, pe linii distincte, cuvintele obţinute prin ştergerea succesivă a vocalelor din cuvântul citit, de la stânga la dreapta, ca în exemplu de mai jos:

Exemplu: Dacă se citeşte cuvântul **examen** se afişează:

xamen
xmen
xmn

11. Scrieţi programul **Pascal/C/C++** care citeşte de la tastatură un cuvânt de cel mult **15** litere mici ale alfabetului englez şi care scrie pe ecran, pe linii distincte, cuvintele obţinute prin ştergerea succesivă a vocalelor în ordinea alfabetică a lor (**a, e, i, o, u**). La fiecare pas se vor şterge toate apariţiile din cuvânt ale unei vocale (ca în exemplu).

Exemplu: Dacă se citeşte cuvântul **bacalaureat** se afişează:

bcluret (s-au şters toate cele patru apariţii ale vocalei **a**)

bclurt (s-a şters unica apariţie a vocalei **e**)

bclrt (s-a şters unica apariţie a vocalei **u**)

12. Scrieţi programul **PASCAL/C/C++** care citeşte de la tastatură un cuvânt **cuv1** având cel mult **10** caractere, construieşte cuvântul **cuv2** format astfel: prima literă a cuvântului **cuv1** scrisă o singură dată, a doua literă a cuvântului **cuv1** repetată de două ori, a treia literă a cuvântului **cuv1** repetată de trei ori, etc… şi afişează pe ecran cuvântul **cuv2**.

Exemplu: pentru **cuv1=mama** se va afişa **cuv2 :maammmaaaa**

13. Să se scrie programul **PASCAL/C/C++** care citeşte de la tastatură un cuvânt de maximum **20** de litere şi minimum o literă şi şterge litera din mijloc în cazul în care cuvântul are un număr impar de litere, respectiv cele două litere din mijlocul cuvântului dacă acesta are un număr par de litere. Programul va afişa cuvântul rezultat în urma ştergerii sau mesajul **CUVANT VID** dacă după ştergere acesta rămâne fară nici o literă.

Exemplu : dacă se citeşte cuvântul **carte**, se va afişa **cate.**

14. Se citesc de la tastatură **n** propoziţii **(0<n<101)**, având fiecare maximum **255** de caractere. Ştiind că oricare două cuvinte consecutive dintr-o propoziţie sunt despărţite printr-un singur spaţiu şi că fiecare propoziţie se termină cu **Enter**, să se afişeze pe ecran propoziţia care are cele mai multe cuvinte. Dacă două sau mai multe propoziţii au acelaşi număr de cuvinte se va afişa prima dintre ele, în ordinea citirii.

a) Explicaţi în limbaj natural metoda de rezolvare utilizată.
b) Scrieţi programul **Pascal** corespunzător.

De exemplu, pentru **n = 3** şi următorele propoziţii:
Azi sunt inca elev.

Maine am examen de bac.

Ura, voi fi student!

Se va afişa

Maine am examen de bac.

15. Scrieţi un program care citeşte de la tastatură două cuvinte **u** şi **v** formate din cel mult **100** de litere mari ale alfabetului englez. Programul va verifica dacă **u** şi **v** au acelaşi număr de caractere şi, în caz afirmativ, va determina un şir de lungime minimă format din cuvinte, astfel încât primul cuvânt din acest şir să coincidă cu **u**, ultimul cu **v** şi orice două cuvinte succesive să difere prin exact o literă. Fiecare cuvânt din acest şir va fi scris pe câte o linie, în fişierul text **TRANS.TXT**. În cazul în care **u** şi **v** nu au acelaşi număr de litere, se va afişa pe ecran mesajul **EROARE**.

De exemplu, dacă **u=MARE** şi **v=TORT**, atunci fişierul **TRANS.TXT** poate să conţină:
MARE

MART

TART

TORT

2.4. Tipul înregistrare

2.4.1. Teste grilă (Pascal | C / C++)

1. Ştiind că s-a făcut una dintre definiţiile de tip alăturate, care din următoarele construcţii este o declarare corectă pentru un tablou cu 10 elemente de tip elev?

Pascal
```
Type elev= record
 nume:string[30];
 nota:real
end;
```

C
```
typedef      struct
elev{
 char nume[30];
 float nota;};
```

C++
```
struct elev{
char nume[30];
float nota;};
```

a. `Var elev : array[1..10] of x;`
b. `Var x[1..10] : array of elev;`
c. `Var x: elev[1..10];`
d. `Var x : array[1..10] of elev;`

a. `struct elev[10];`
b. `struct x elev[10];`
c. `x elev[10];`
d. `struct elev x[10];`

2. Care dintre următoarele reprezintă o declarare corectă pentru o variabilă **x** care memorează simultan numele şi vârsta a maximum 30 de elevi?

a.
```
Type x=record nume:string[30];
                varsta:byte
        end;
```

b.
```
Var x.nume:string[30];
 x.varsta:array[1..30] of byte;
```

c.
```
Var x:record nume:string;
      varsta:byte end;
```

d.
```
Var x:array[1..30] of record
    nume:string; varsta:byte
                      end;
```

a.
```
typedef struct
      {char nume[30];
       int varsta;} x;
```

b.
```
char x.nume[30];
int x.varsta[30];
```

c.
```
struct {char nume[30];
         int varsta;} x;
```

d.
```
struct {char nume[30];
        int varsta;}x[30];
```

3. Ştiind că variabila **p** este utilizată pentru a memora coordonatele reale ale unui punct în plan, care dintre următoarele declarări este corectă?

a. `var p:record x,y:real end;`
b.
```
var p:record
      x,y,z:integer end;
```
c. `real p;`
d. `double p;`

a. `struct punct{float x,y;} p;`
b. `struct punct{int x,y,z;} p;`
c. `float p;`
d. `double p;`

4. Ştiind că s-au făcut declarările alăturate, stabiliţi care dintre următoarele expresii este corectă din punct de vedere sintactic?

```
Type elev = record

Nume:string[30];nota:real
            end;
Var a : array[1..100] of elev;
```

```
struct elev {
   char nume[30];
   float nota;} a[100];
```

a. `elev[1].nota`
b. `a[1].nota[1]`
c. `a.nota[1]`
d. `a[1].nota`

5. Ştiind că fiecare dintre variabilele **e1**, **e2** conţine numele şi media unui elev în forma dată de declararea alăturată, indicaţi care dintre următoarele expresii atribuie variabilei reale **m** media aritmetică a mediilor celor doi elevi:

```
type elev = record
     nume:string[30];
     media:real;  end;
var e1,e2:elev;
```

a. `m:=media(e1+e2)/2`
b. `m:=e1.media+e2.media/2`
c. `m:=(e1+e2).media/2`
d. `m:=(e1.media+e2.media)/2`

```
struct elev
{ char nume[30];
  float media;
} e1,e2;
```

a. `m=media(e1+e2)/2;`
b. `m=e1.media+e2.media/2;`
c. `m=(e1+e2).media/2;`
d. `m=(e1.media+e2.media)/2;`

6. Considerându-se declarările alăturate şi presupunând că în câmpul **latura** al variabilei **p** este reţinută o valoare ce reprezintă latura unui pătrat, care din următoarele expresii atribuie câmpului **aria** al variabilei **p** valoarea ariei pătratului respectiv?

```
type patrat=record
        latura,aria:real;
            end;
var p:patrat;
```

a. `aria:=p.latura*p.latura;`
b. `p.aria:=p.latura*p.latura;`
c. `aria:=latura*latura;`
d. `p.aria:=latura*latura;`

```
struct patrat
{
  float latura,aria;
}p;
```

a. `aria=p.latura*p.latura;`
b. `p.aria=p.latura*p.latura;`
c. `aria=latura*latura;`
d. `p.aria=latura*latura;`

7. Se defineşte tipul

```
punct=record x,y:real end;
```

```
struct punct {float x,y;};
```

(pentru a memora coordonatele carteziene - abscisa şi ordonata - ale unui punct din planul **xOy**) şi tabloul unidimensional **a** cu elemente de tip **punct**. Care dintre expresiile de mai jos are valoarea **true(1)** dacă şi numai dacă punctul ale cărui coordonate sunt memorate în elementul din tablou aflat pe poziţia **i** are abscisa şi ordonata egale?

a. `a[i]^.x=a[i]^.y`

a. `a[i]->x==a[i]->y`

b. `x[a[i]]=y[a[i]]`	b. `x[a[i]]==y[a[i]]`
c. `a[i].x=a[i].y`	c. `a[i].x==a[i].y`
d. `a.x[i]=a.y[i]`	d. `a.x[i]==a.y[i]`

8. Fie declarările de mai jos. Dacă variabila **x** reţine informaţii despre **30** de elevi, precizaţi care este varianta corectă ce afişează numele şi media elevului al **11**-lea?

```
Type elev=record
             nume:string[30];
             media:real
             end;
var x:array[1..30] of elev;
```

```
struct elev{
       char nume[30];
       float media; };
elev x[30];
```

Varianta Pascal	a. `writeln(x[11].nume,' ',x[11].media);`
	b. `writeln(x.nume,' ',x.media);`
	c. `writeln(x.nume[11],' ',x.media);`
	d. `writeln(x[11]^.nume,' ', x[11]^.media);`
Varianta C/C++	a. `cout<<x[10].nume<<" "<<x[10].media;` / `printf("%s %f", x[10].nume,x[10].media);`
	b. `cout<<x.nume<<" "<<x.media;` / `printf("%s %f", x.nume,x.media);`
	c. `cout<<x.nume[11]<<" "<<x.media;` / `printf("%s %f", x.nume[11],x.media);`
	d. `cout<<x[10]->nume<<" "<< x[10]->media);` / `printf("%s %f", x[10]->nume,x[10]->media);`

9. Variabila **P**, de tip înregistrare, memorează în două câmpuri **x** şi **y** două numere reale reprezentând coordonatele punctului **P** din plan. Condiţia ca acest punct să coincidă cu unul dintre punctele **(1,0)** sau **(0,1)** este:

Varianta Pascal	a. `(P.x=0) and (P.y=1) and (P.x=1) and (P.y=0)`
	b. `(P.x=0) and (P.y=1) or (P.x=1) and (P.y=0)`
	c. `(P.x=0) or (P.y=1) and (P.x=1) or (P.y=0)`
	d. `(P.x=0) or (P.y=1) or (P.x=1) or (P.y=0)`
Varianta C/C++	a. `P.x==0 && P.y==1 && P.x==1 && P.y==0`
	b. `P.x==0 && P.y==1 || P.x==1 && P.y==0`
	c. `P.x==0 || P.y==1 && P.x==1 || P.y==0`
	d. `P.x==0 || P.y==1 || P.x==1 || P.y==0`

10. Se consideră declararea

```
type fractie =
 record x,y:integer;end
```

```
struct fractie
{unsigned int x,y;}s,f1,f2;
```

şi variabilele **s,f1,f2** de acest tip în care câmpurile **x** şi **y** reprezintă numărătorul, respectiv numitorul unei fracţii. Care dintre următoarele variante construieşte în variabila **s** o fracţie echivalentă cu suma fracţiilor **f1** şi **f2**?

a.
```
s.x:=f1.x+f2.x;
s.y:=f1.y+f2.y
```

b.
```
s.x:=f1.x*f2.y+f1.y*f2.x;
s.y:=f1.y*f2.y
```

c.
```
s:=f1+f2
```

d.
```
s.x:=f1.x*f1.y+f2.x*f2.y;
s.y:=f1.y*f2.y
```

a.
```
s.x=f1.x+f2.x;
s.y=f1.y+f2.y
```

b.
```
s.x=f1.x*f2.y+f1.y*f2.x;
s.y=f1.y*f2.y
```

c.
```
s=f1+f2
```

d.
```
s.x=f1.x*f1.y+f2.x*f2.y;
s.y=f1.y*f2.y
```

11. Considerăm următoarea declarare:

```
type produs =record
      denumire:string[10];
      pret:integer
            end;
var p:produs;
```

```
struct produs
{
  char denumire[10];
  int pret;
 } p;
```

Cum se poate accesa prima literă a denumirii unui produs ale cărui caracteristici sunt memorate în variabila p ?

a. `p.denumire[0]`
b. `p^denumire`
c. `p.denumire[1]`
d. `p^denumire[1]`

a. `p.denumire[1]`
b. `p->denumire`
c. `p.denumire[0]`
d. `p->denumire[1]`

12. Pentru a memora numele şi vârsta unei persoane în variabila **x**, se utilizează declararea:

a.
```
var x.nume:string;
    x.varsta:byte;
```

b.
```
var x:byte;
```

c.
```
var x:record a:string;
    b:byte end;
```

d.
```
var x:record nume:real;
             varsta:byte end;
```

a.
```
char x.nume[40];
int x.varsta;
```

b.
```
int x;
```

c.
```
struct
 {char a[40]; int b;} x;
```

d.
```
struct x
 {float nume; int varsta;}
```

13. Considerăm următoarea declarare:

```
type persoana=record
        nume,prenume:string[10];
        varsta:integer;
              end;
var p:persoana;
```

```
struct persoana
  { char nume[10];
    char prenume[10];
    int varsta;
  } p;
```

Cum se poate accesa prima literă a numelui unei persoane ale cărei date de identificare sunt memorate în variabila **p** ?

a. `p^nume[1]`
b. `p^nume`
c. `p.nume[1]`
d. `p.nume[0]`

a. `p->nume[1]`
b. `p->nume`
c. `p.nume[0]`
d. `p.nume[1]`

14. Variabila **elev** este utilizată pentru a memora numele unui elev şi media sa la bacalaureat. Declararea corectă a variabilei **elev** este:

a.
```
var elev:string[60];
```
b.
```
var elev:record
   nume:string[30];medie:real
         end;
```
c.
```
var elev:array[1..2]of real;
```
d.
```
type elev=record
   nume:string[30];medie:real
end;
```

a.
```
char elev[60];
```
b.
```
struct { char nume[30];
         float medie;}elev;
```
c.
```
float elev[2];
```
d.
```
struct elev {char nume[30];
             float medie;};
```

15. Pentru următoarele declarări:

```
type elev=record
  nume : string[19];
  nr : integer;
  nota:array[1..14]of integer
end;
var e:array[1..27] of elev;
    x:elev;
```

```
struct elev
   { char nume[20];
     int nr, nota[15];
   };
elev e[28],x;
```

Care dintre următoarele instrucţiuni este corectă?

a. `x.nota:= x.nota+1;`
b. `e[10]:= x.nr;`
c. `e[10]:= x;`
d. `elev.nota[5]:= 7;`

a. `x.nota=x.nota+1;`
b. `e[10]=x.nr;`
c. `e[10]=x;`
d. `elev.nota[5]=7;`

16. Într-un vector sunt memorate înregistrări. Fiecare înregistrare reţine numele şi media notelor unui elev la examenul de bacalaureat. Vectorul are 1000 de elemente şi este ordonat după numele elevilor. Care din următoarele metode este mai eficientă pentru a găsi media notelor unui elev al cărui nume îl cunoaştem ?

a. **căutarea binară**
b. **căutarea secvenţială**
c. **Backtracking**
d. **interclasarea**

17. Dacă în cadrul unui program variabila **p** urmează să memoreze simultan coordonatele reale (abscisa şi ordonata) pentru un punct în plan, atunci variabila **p** trebuie declarată astfel:

a. `Var p:record x,y:real;end;`	**a.** `struct punct{float x,y;} p;`
b. `Var p:string[2];`	**b.** `char p[2];`
c. `Type p=record x,y:real; end;`	**c.** `typedef struct p{float x,y;};`
d. `Var p:real;`	**d.** `float p;`

18. Variabila **e** este utilizată pentru a memora următoarele date referitoare la un elev ce participă la examenul de bacalaureat: **numele, prenumele** şi **media** obţinută la examen. **Numele** şi **prenumele** conţin fiecare cel mult **15** caractere iar media este o valoare reală care urmează să fie utilizată în diverse calcule matematice. Stabiliţi care dintre următoarele declarări sunt corecte şi folosesc în mod eficient spaţiul de memorie:

Varianta Pascal	**a.**	`var e:array[1..15,1..15] of char;`

	b.	`type elev= record nume:char;`
		`prenume: string[15];media:real end;`
		`var e:elev;`

	c.	`type elev=record nume:string[15];`
		`prenume:string[15];media[5]:real end;`
		`var e:elev;`

	d.	`type elev=record`
		`nume,prenume:string[15];media:real end;`
		`var e:elev;`

Varianta C/C++	**a.**	`char e[15][15];`
	b.	`struct{char nume,prenume[15];float media;} e;`
	c.	`struct{char nume[16],prenume[16];float media[5];} e;`
	d.	`struct{char nume[16],prenume[16];float media;} e;`

19. Variabila **b** reţine în câmpul **fraza** un text, format din cel mult 255 de caractere şi în câmpul **nrl** numărul de litere mici din text. Care dintre referirile următoare reprezintă primul caracter din câmpul **fraza** al variabilei **b**?

```
type  a=record                    struct
      fraza:string;nrl:integer;   {char fraza[256];
      end;                         int nrl;} b;
var b:a;
```

a. `b.fraza[1]`	**a.** `b.fraza[0]`
b. `b.fraza`	**b.** `b.fraza`
c. `b[1].fraza[1]`	**c.** `b[0].fraza[0]`
d. `b[1].fraza`	**d.** `b[0].fraza`

2.4.2. Probleme

1. Scrieţi un program **PASCAL/C/C++** care citeşte de pe prima linie a fişierului **BAC.TXT** un număr natural **n** şi de pe următoarele **n** linii, câte două numere reale reprezentând coordonatele carteziene (abscisă, ordonată) a **n** puncte din planul **xOy**. Programul afişează pe ecran coordonatele punctelor care se află la cea mai mică distanţă de originea O a planului **xOy**; coordonatele fiecărui astfel de punct vor fi separate prin virgulă şi grupate între paranteze rotunde.

Exemplu: dacă în fişier se află numerele

```
3
2.0 1.0
2.0 2.0
1.0 2.0
```

Se afişează: **(2.0,1.0) (1.0,2.0)**

2. Scrieţi programul **PASCAL/C/C++** care citeşte un număr natural **n** (**0<n<30**), apoi citeşte numele şi media generală a fiecăruia dintre cei **n** elevi ai unei clase, mediile generale fiind valori diferite două câte două. Să se afişeze numele şi media celor **n** copii, în ordinea descrescătoare a mediilor. Datele fiecărui copil se vor afişa pe câte o linie, numele şi media separate printr-un spaţiu, media având două cifre la partea zecimală.

3. Scrieţi un program Pascal/C/C++ care citeşte un număr natural nenul **n** **(n<100)** şi un şir de **n** numere naturale nenule de cel mult **4** cifre fiecare, şi care afişează pe ecran şirul ordonat crescător în funcţie de suma cifrelor corespunzătoare fiecărui termen al său. Dacă două numere au aceeaşi sumă a cifrelor, se va afişa mai întâi cel mai mic dintre ele.

De exemplu, pentru **n=5** şi numerele **701,1000,44,99,143,** se va afişa sirul:

1000 44 143 701 99

3. Fişiere

1. Scrieţi un program care citeşte de la tastatură un număr natural nenul **n** (**n≤20**) şi construieşte fişierul **stelute.txt** astfel încât acesta să conţină pe prima linie un caracter *****, pe a doua linie două caractere *****, …, pe a **n**–a linie **n** caractere *****, pe linia **n+1**, **n-1** caractere *****, pe linia **n+2**, **n-2** caractere *****,…, iar pe linia **2n-1** un caracter *****.

De exemplu, dacă **n=4** fişierul are conţinutul:

```
*
**
***
****
***
**
*
```

2. Scrieţi programul **C/C++** care citeşte de la tastatură un număr natural **n(0<n<100)** şi crează fişierul text **BAC.TXT** care conţine pe prima linie, în ordine descrescătoare, toate numerele naturale de la **n** la **1**, pe a doua linie în ordine descrescătoare, toate numerele naturale de la **n-1** la **1** etc... pe linia **n-1** numerele **2 1**, iar pe ultima linie numărul **1**. Pe fiecare linie numerele vor fi despărţite prin câte un spaţiu.
Exemplu: dacă se citeşte **n=3,** atunci **BAC.TXT** va conţine:

```
3 2 1
2 1
1
```

3. Scrieţi programul **Pascal/C/C++** care citeşte de la tastatură un număr natural **n(0<n<100)** şi creează fişierul text **BAC.TXT** care conţine pe prima linie numărul **1**, pe a doua linie numerele **2 1** , pe a treia linie, în ordine descrescătoare, numerele **3 2 1** etc., pe ultima linie, în ordine descrescătoare, toate numerele naturale de la **n** la **1**. Pe fiecare linie numerele vor fi despărţite prin câte un spaţiu.
De exemplu, dacă se citeşte **n=3,** atunci **BAC.TXT** va conţine:

```
1
2 1
3 2 1
```

4. Scrieţi programul Pascal/C/C++ care, pentru o valoare **n** (număr natural, **1<=n<=100**) citită de la tastatură, scrie în fişierul **bac.txt**, separate prin spaţiu, primele **n** numere strict pozitive divizibile cu **3**.
De exemplu, pentru **n=4**, conţinutul fişierului este : **3 6 9 12**

5. Fişierul **bac.txt** conţine pe primul rând un număr natural nenul **n** cu cel mult cinci cifre şi pe fiecare dintre următoarele **n** linii câte două numere naturale **a, b** (**a<b**) cu cel mult **3** cifre fiecare, separate printr-un spaţiu, numere ce reprezintă capetele unui interval închis. Se cere să se afişeze pe ecran, separate printr-un spaţiu, două numere **x** şi **y** ce reprezintă capetele intervalului de intersecţie a celor **n** intervale date. Dacă intersecţia lor este mulţimea vidă, se va afişa mesajul **multime vida.**

De exemplu, dacă fişierul **bac.txt** conţine: se afişează

```
4                                   8 10
1 10
5 20
8 12
1 21
```

a) Alegeţi o metodă de rezolvare care să utilizeze eficient spaţiul de memorie, descriind în limbaj natural metoda folosită şi justificând eficienţa acesteia (cel mult 6 rânduri)

b) Scrieţi programul Pascal/C/C++ corespunzător metodei descrise la punctul a).

6. Scrieţi un program care creează fişierul text **bac.txt** ce conţine o linie unică având în ordine descrescătoare, toate numerele pare mai mici sau egale cu **n**, unde **n** este un număr natural citit de la tastatură (**n<1000**). Numerele scrise în fişier vor fi separate prin câte un spaţiu. De exemplu, dacă se citeşte **n=11**, atunci **bac.txt** va conţine: **10 8 6 4 2 0**.

7. Fişierul **numere.in** conţine pe mai multe linii câte două valori numerice naturale de maximum patru cifre fiecare, separate printr-un spaţiu. Să se scrie în fişierul **numere.out** un număr egal de linii cu cel din fişierul **numere.in**, linii care vor avea următorul conţinut:

- prima linie din **numere.out** va conţine cea mai mică valoare de pe prima linie a fişierului **numere.in** urmată de media aritmetică a celor două valori de pe prima linie a fişierului **numere.in** urmată de cea mai mare dintre cele două valori aflate pe prima linie a fişierului **numere.in** ; între aceste trei valori se va lăsa un spaţiu, iar media aritmetică va fi afişată cu exact o zecimală;
- a doua linie din **numere.out** va conţine cea mai mică valoare de pe a doua linie a fişierului **numere.in** urmată de media aritmetică a celor două valori de pe a doua linie a fişierului **numere.in** urmată de cea mai mare dintre cele două valori aflate pe a doua linie a fişierului **numere.in** ; între aceste trei valori se va lăsa un spaţiu, iar media aritmetică va fi afişată cu exact o zecimală, etc.

Exemplu:

numere.in	**numere.out**
2 7	**2 4.5 7**
200 300	**200 250.0 300**
1001 1000	**1000 1000.5 1001**
5 5	**5 5.0 5**

8. Fişierul text **BAC.IN** are următoarea structură: pe prima linie se găseşte un număr natural **n** (**2≤n≤1.000.000**), iar pe a doua linie se găsesc **n** numere întregi formate din cel mult **9** cifre şi despărţite prin spaţiu. Ştiind că fişierul **BAC.IN** conţine întotdeauna cel puţin două numere distincte printre cele **n** aflate pe cea de-a doua linie, scrieţi un program care să creeze mai întâi fişierul text **BAC.OUT** şi apoi să scrie în el, pe prima linie, separate printr-un

spaţiu şi în ordine descrescătoare, cele mai mari două valori distincte din fişierul **BAC.IN**. Alegeţi o metodă eficientă de rezolvare atât ca timp de executare, cât şi ca gestionare a memoriei.
Exemplu:

BAC.IN	**BAC.OUT**
10	**11 8**
5 11 7 3 8 -2 11 11 -7 5	

9. Pe prima linie a fişierului text **BAC.TXT** se află un număr natural nenul **n** format din maximum **6** cifre, iar pe a doua linie a fişierului se află un şir format din **n** numere naturale nenule, despărţite prin spaţii, fiecare număr fiind format din cel mult **4** cifre. Să se afişeze lungimea maximă a unei secvenţe strict crescătoare din şirul dat. O secvenţă a unui şir constă în elemente aflate pe poziţii consecutive în şirul considerat. Alegeţi o metodă eficientă de rezolvare atât ca timp de executare, cât şi ca gestionare a memoriei.

Dacă, de exemplu, fişierul **BAC.TXT** are conţinutul
10
7 22 13 **10 31 41** 6 **2 10 18**
atunci programul trebuie să afişeze valoarea **3** deoarece în şirul considerat există mai multe secvenţe care au proprietatea cerută, dar lungimea maximă a unei astfel de secvenţe este **3**.

10. Pe prima linie a fişierului text **BAC.TXT** se află un număr natural nenul **n** (**2≤n<1000000**), iar pe a doua linie a fişierului se află un şir format din **n** numere întregi, despărţite prin câte un spaţiu, fiecare număr fiind format din cel mult **4** cifre. Ştiind că în fişier există cel puţin un număr strict pozitiv, se cere să se afişeze lungimea maximă a unei secvenţe din şir care are proprietatea că este formată doar din valori strict pozitive. O secvenţă a unui şir este formată dintr-unul sau mai multe elemente aflate pe poziţii consecutive. Alegeţi o metodă eficientă de rezolvare atât ca timp de executare, cât şi ca gestionare a memoriei.

De exemplu, dacă fişierul **BAC.TXT** are conţinutul
10
7 22 -3 10 3 14 0 21 10 8
atunci programul trebuie să afişeze pe ecran valoarea **3** deoarece în şirul considerat există mai multe secvenţe care sunt formate doar din numere strict pozitive, dar lungimea maximă a unei astfel de secvenţe este **3**.

11. Pe prima linie a fişierului **bac.txt** se află cel mult **1000** de numere întregi de maximum trei cifre fiecare, separate prin câte un spaţiu. Scrieţi un program care construieşte fişierul **cifre.txt** astfel încât acesta să conţină pe prima linie, separate prin câte un spaţiu, toate numerele de exact două cifre din fişierul **bac.txt**.
Exemplu: Dacă în fişierul **bac.txt** se găsesc numerele:
100 34 1 78 90 123 0 67 8 -90 -899 -9 88
fişierul **cifre.txt** va conţine numerele: **34 78 90 67 -90 88**

12. Fişierul text **BAC.TXT** conţine **100** de numere naturale de cel mult **6** cifre fiecare, câte un număr pe fiecare linie a fişierului. Scrieţi un program care:

a) afişează pe ecran toate numerele din fişier, câte **5** pe fiecare linie, numerele de pe o linie fiind separate separate prin câte un spaţiu;

b) afişează pe ecran suma numerelor prime din fişier.

13. Scrieţi programul Pascal/C/C++ care citeşte din fişierul **BAC.TXT** un şir **s** de cel mult un milion de numere naturale, fiecare număr având cel mult patru cifre, şi care determină în mod eficient din punctul de vedere al timpului de executare şi al spaţiului de memorie folosit, numărul de componente ale şirului obţinut prin eliminarea din cele două extremităţi ale lui **s** a unui număr minim de componente, astfel încât şirul rezultat să înceapă şi să se termine cu câte un număr par. Fişierul **BAC.TXT** conţine cel puţin un număr par iar numerele din fişier sunt separate printr-un singur spaţiu. Programul va afişa pe ecran numărul de componente ale şirului obţinut.

De exemplu, dacă fişierul **BAC.TXT** conţine numerele:

1 245 22 67 34 29 345 8 354 11 7 34 12 45 39 41 26 67 89 1011

se va afişa pe ecran numărul: **15,** deoarece sunt eliminate numerele subliniate iar şirul rezultat este format din **15** numere.

14. Scrieţi programul Pascal/C/C++ care citeşte din fişierul **BAC.TXT** un şir **s** de cel mult un milion de numere naturale, fiecare număr având cel mult patru cifre, şi care determină în mod eficient din punctul de vedere al timpului de executare şi al spaţiului de memorie folosit, numărul de componente ale şirului obţinut prin eliminarea din cele două extremităţi ale lui **s** a unui număr minim de componente, astfel încât şirul rezultat să înceapă şi să se termine cu câte un număr par. Fişierul **BAC.TXT** conţine cel puţin un număr par iar numerele din fişier sunt separate printr-un singur spaţiu. Programul va afişa pe ecran numărul de componente ale şirului obţinut.

De exemplu, dacă fişierul **BAC.TXT** conţine numerele:

1 245 22 67 34 29 345 8 354 11 7 34 12 45 39 41 26 67 89 1011

se va afişa pe ecran numărul: **15,** deoarece sunt eliminate numerele subliniate iar şirul rezultat este format din **15** numere.

15. Fişierul **DATE.IN** conţine cel mult **100000** numere naturale separate prin spaţii, fiecare număr având cel mult nouă cifre. Să se realizeze un program Pascal/C/C++ care scrie în fişierul **DATE.OUT,** pe o singură linie, separate prin câte un spaţiu, toate numerele din **DATE.IN** care au prima cifra pară. Prima cifra a unui număr este cifra cea mai din stânga (adică cea mai semnificativă cifră).

Exemplu: dacă fişierul **DATE.IN** conţine:

45 123 68 8 134 56 876 6666 2 5 123 65
fişierul **DATE.OUT** va avea următorul conţinut: **45 68 8 876 6666 2 65**

16. În fişierul **bac.txt** pe aceeaşi linie, despărţite printr-un spaţiu, se găsesc maximum **1000** de numere întregi. Scrieţi un program care să determine maximul şi minimul numerelor din fişier folosind o metodă eficientă din punct de vedere al memoriei şi al numărului de operaţii efectuate şi afişează pe ecran valoarea maximului şi a minimului.
Exemplu: Dacă în fişierul **bac.txt** se găsesc numerele **100 34 1 78 90 123 0 67 8 -90 -899 -9 88** se va afişa **max=123** şi **min=-899**.

17. Se consideră şirul **crescător** format din toate numerele naturale nenule, în care fiecare număr apare de exact **k** ori. De exemplu, pentru **k=3** termenii şirului sunt **1,1,1,2,2,2,3,3,3,4,4,4,5,5,5...**

a) Scrieţi un program **Pascal/C/C++** ce citeşte două numere naturale, **k** şi **n** (**1≤k≤100, 1≤n≤1000000000**) şi, utilizând un algoritm eficient din punct de vedere al spaţiului de memorare şi al timpului de executare, scrie în fişierul **BAC.TXT** primii **n** termeni ai şirului (separaţi prin câte un spaţiu), în ordine descrescătoare.

Exemplu: pentru **k=3** şi **n=10** în fişier se scriu în ordine, numerele:

4 3 3 3 2 2 2 1 1 1 (deoarece primii **10** termeni ai şirului sunt **1,1,1,2,2,2,3,3,3,4)**

b) Descrieţi în limbaj natural metoda utilizată şi explicaţi în ce constă eficienţa ei.

18. Fişierul **numere.in** conţine cel mult **1000** de numere întregi separate prin spaţii. Numerele din fişier au valori cuprinse intre **-30000** şi **30000**.

Să se determine cel mai mare număr din fişier precum şi numărul de apariţii ale acestuia.

Exemplu: dacă fişierul conţine numerele: **2 7 12 3 8 12 9 5** atunci maximul este **12** şi acesta apare de două ori în fişier.

Alegeţi un algoritm eficient de rezolvare.

a) Explicaţi în limbaj natural metoda utilizată, justificând eficienţa ei (4-6 rânduri).

b) Scrieţi programul **Pascal/C/C++** ce rezolvă problema enunţată şi afişează pe ecran cele două valori cerute. În rezolvarea problemei se va utiliza metoda descrisă la punctul anterior.

19. Se citeşte de la tastatură un număr natural **n** (**0<n<100**). Scrieţi programul **Pascal/C/C++** ce construieşte fişierul **prime.out** cu primele **n** numere prime, în ordine crescătoare, separate prin câte un spaţiu.

Exemplu: Pentru **n=8**, fişierul **prime.out** conţine: **2 3 5 7 11 13 17 19**

20. Pe prima linie a fişierului text **BAC.IN** se află un număr natural **n** (**0 < n <= 1000**), iar pe a doua linie **n** numere reale pozitive, despărţite prin câte un spaţiu.

Scrieţi programul **Pascal** care citeşte datele din fişierul de intrare şi scrie în fişierul text **BAC.OUT** pe o linie, despărţite prin câte un spaţiu, numerele care au partea întreagă număr prim. Dacă nici unul din numere nu are partea întreagă număr prim atunci se va scrie în fişierul **BAC.OUT** mesajul **NU EXISTA**

De exemplu, dacă **BAC.IN** conţine
6
12.095 31.567 5.789 789.834 1234.923 2.345
atunci fişierul **BAC.OUT** va conţine
31.567 5.789 2.345

21. În fişierul text **NUMERE.IN,** pe prima linie se află trei numere naturale **n**, **a** şi **b (0<n<=1000, a<b)** separate prin câte un spaţiu, iar pe linia a doua **n** numere naturale despărţite prin câte un spaţiu. Scrieţi programul care citeşte datele din fişierul **NUMERE.IN** şi scrie în fişierul text **NUMERE.OUT** numerele de pe linia a doua a fişierului de intrare care se găsesc în afara intervalului deschis **(a,b)**. Numerele vor fi scrise pe o singură linie în fişierului de ieşire separate prin câte un spaţiu.

22. Fişierul text **BAC.IN** conţine pe prima linie un număr natural **n (0<n<=5000)**, iar pe a doua linie **n** numere naturale de cel mult **9** cifre fiecare, separate prin câte un spaţiu. Să se scrie un program care citeşte **n**, apoi cele **n** numere naturale din fişierul **BAC.IN** şi scrie în fişierul text **BAC.OUT**, pe câte o linie fiecare, numerele de pe a doua linie a fişierului **BAC.IN** care au exact patru cifre.

Exemplu:

BAC.IN	BAC.OUT
6	1124
23 1124 567 89013 5586 1200	5586
	1200

23. Din fişierul text **BAC.IN** se citeşte un număr natural **n** de cel mult 9 cifre. Scrieţi un program **Pascal** care scrie în fişierul text **BAC.OUT**, pe o singură linie, separaţi prin câte un singur spaţiu, toţi divizorii de trei cifre ai numărului

n. Dacă numărul **n** nu are divizori formaţi din trei cifre, atunci se va scrie în fişier mesajul **NU EXISTA**.

De exemplu, pentru **n = 1000** se va scrie în **BAC.OUT 100 125 200 250 500**

Iar pentru **n = 12347 BAC.OUT** va conţine **NU EXISTA**.

24. Pentru un număr natural **n** citit de a tastatură (**0<n<100**) se cere să se construiască un vector cu n componente numere naturale din mulţimea **{0, 1, 2}** astfel încât să nu existe două elemente egale alăturate şi suma oricăror **3** elemente consecutive să fie egală cu **3**. Scrieţi programul Pascal care generează în memorie vectorul şi apoi scrie în fişierul text **VECT.TXT** componentele vectorului, în ordine, de la prima până la ultima, cu spaţii între orice două elemente consecutive.

De exemplu, pentru **n=4**, fişierul **VECT.TXT** poate să conţină valorile **1 2 0 1**

25. Prin secţiune a unui şir **V** înţelegem o succesiune de elemente ale lui **V** situate pe poziţii consecutive în şirul dat. Dacă în fişierul text **bac.in** se află pe prima linie un număr natural **n(1<n<10000)** iar pe linia următoare cele **n** componente întregi ale şirului **V**, componente cu cel mult patru cifre fiecare, se cere să se determine în mod eficient secţiunea de sumă minimă. Se va afişa pe ecran numărul reprezentând suma minimă determinată.

a) Explicaţi în limbaj natural metoda utilizată, justificând eficienţa acesteia (4-5 rânduri).
b) Scrieţi programul corespunzător metodei descrise la punctul a).

De exemplu, dacă fişierul **bac.in** conţine:

6
-3 2 -3 -4 5 6
se va afişa : **-8**

26. Fişierul **numere.in** conţine mai multe numere naturale, mai mici decât **1000**. Acestea sunt scrise pe o singură linie şi sunt separate prin spaţiu. Scrieţi programul **Pascal/C/C++** ce afişează pe ecran numărul de valori din fişier.

27. Se consideră două fişiere text **F1.TXT** şi **F2.TXT** ce conţin numere reale dispuse pe mai multe linii şi separate prin spaţii. Fişierul **F1.TXT** conţine numai numere distincte, iar fişierul **F2.TXT** conţine numai numere din fişierul **F1.TXT**, dar acestea se pot repeta.

Scrieţi programul PASCAL/C/C++ care afişează pentru fiecare valoare din fişierul **F1.TXT** numărul său de apariţii în fişierul **F2.TXT.** Pe linii diferite ale ecranului vor fi afişate câte două valori, separate prin spaţiu, prima fiind numărul din fişierul **F1.TXT,** iar a doua numărul său de apariţii din **F2.TXT.**

28. În fişierul text **BAC.TXT** se află mai multe valori numerice întregi de cel mult nouă cifre dispuse pe mai multe linii şi separate prin spaţii. Să se scrie programul PASCAL/C/C++ care afişează pe ecran câte valori din fişierul **BAC.TXT** sunt situate strict între **100** şi **10000**.

29. Se citeşte de la tastatură un număr natural nenul **n** care are cel mult 9 cifre. Să se afişeze în fişierul **Date.out** numărul **k**, natural, astfel încât produsul **1*2*3*...*(k-1)*k** să aibă o valoare cât mai apropiată de numărul **n**.

De exemplu, dacă se citeşte numarul **n=25** fişierul **Date.out** are următorul conţinut:

4

De exemplu dacă se citeşte numărul **n=119** fişierul **Date.out** are următorul conţinut:

5

30. Se citeşte de la tastaură un număr natural nenul **n (n<1000)**. Scrieţi programul **PASCAL/C/C++** care construieşte fişierul text **bac.txt** care să conţină, pe prima linie, toţi divizorii lui **n** în ordine strict descrescătoare. Divizorii vor fi separaţi prin spaţiu.

De exemplu, dacă **n=10**, atunci fişierul **bac.txt** va conţine:

10 5 2 1

31. Se consideră fişierul text **date.in** care conţine exact 899 de numere distincte formate fiecare din câte 3 cifre. Să se afişeze în fişierul text **date.out** numărul format din 3 cifre care lipseşte din fişierul text **date.in**.

32. Fişierul **bac.txt** conţine un şir de 2007^2 numere naturale cu cel mult **două** cifre fiecare. Se cere determinarea şi afişarea pe ecran, în ordine crescătoare, a valoriilor distincte din fişier.
Numerele sunt afişate câte **5** pe fiecare linie (cu excepţia ultimei linii care poate să conţină mai puţine valori) şi sunt separate prin spaţiu.

De exemplu, dacă fişierul **bac.txt** are conţinutul
3 3 7 22 82 7 56 15 3 56 7 22 82 15 7 23 82 82...82
atunci se va afişa:
3 7 15 22 23
56 82

a) Alegeţi o metodă eficientă de rezolvare, descriind în limbaj natural metoda folosită şi explicaţi eficienţa acesteia (cel mult 6 rânduri)
b) Scrieţi programul Pascal/C corespunzător metodei descrise la punctul a).

33. Un şir crescător de fracţii ireductibile din intervalul **[0,1]**, cu numitorul mai mic sau egal cu **n**, se numeşte şir Farey de ordin **n**, notat F_n. Şirurile Farey de ordin **1, 2** şi **5** conţin elementele:

$F_1 = \{\frac{0}{1}, \frac{1}{1}\}$; $F_2 = \{\frac{0}{1}, \frac{1}{2}, \frac{1}{1}\}$,…,

$F_5 = \{\frac{0}{1}, \frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{2}{5}, \frac{1}{2}, \frac{3}{5}, \frac{2}{3}, \frac{3}{4}, \frac{4}{5}, \frac{1}{1}\}$

Dacă notăm o astfel de serie de valori cu: x_0/y_0, x_1/y_1, …, x_i/y_i,… atunci valorile care apar într-un şir Farey se pot calcula cu următoarele relaţii:

$x_0=0, y_0=1, x_1=1, y_1=n,$

$x_{i+2}=[(y_i+n)/y_{i+1}]x_{i+1}-x_i$, $y_{i+2}=[(y_i+n)/y_{i+1}]y_{i+1}-y_i$,

în care s-a notat cu **[a]** partea întreagă a lui **a**.

Scrieţi un program Pascal/C/C++ care citeşte de la tastatură un număr natural **n** ($1 \le n \le 50$) şi apoi creează fişierul text **numere.txt** care conţine pe fiecare linie, separate prin spaţiu, numărătorul şi numitorul unei fracţii din şirul Farey de ordinul **n**.

De exemplu pentru **n=2**, conţinutul fişierului **numere.txt** va fi:

```
0 1
1 2
1 1
```

34. Prin asocierea fiecărei litere mici din alfabetul englez cu un număr egal cu poziţia literei în alfabet, se poate codifica orice secvenţă de litere mici cu o secvenţă de numere. Scrieţi un program Pascal care citeşte din fişierul text **cod.txt** un număr **n** ($0<n<30000$) şi, de pe rândul următor, o secvenţă de **n** de litere mici. Folosind un algoritm eficient din punctul de vedere al gestionării memoriei, programul va codifica secvenţa de caracterele citită şi va scrie pe ecran numerele asociate, separate prin spaţii.

Exemplu: pentru fişierul **cod.txt**:

```
7
bacinfo
```

se vor afişa pe ecran, în ordine, numerele:
2 1 3 9 14 6 15.

35. Fişierul text **BAC.TXT** conţine o singură linie pe care este scrisă o expresie aritmetică fără paranteze, în care operanzii sunt cifre, iar operatorii sunt **+** sau **-**. La sfârşitul expresiei, în fişier, se află caracterul **=**. Expresia are cel mult **30** de operanzi şi este corectă (primul şi ultimul caracter din expresie sunt cifre, între două cifre există exact un operator, nu există două cifre una lângă alta şi nu conţine alte caractere). Scrieţi programul Pascal/C/C++ care citeşte din fişierul text **BAC.TXT** expresia aritmetică şi determină în mod eficient, din punctul de vedere al spaţiului de memorie folosit, rezultatul evaluării acesteia. Programul va afişa pe ecran atât expresia cât şi rezultatul evaluării, pe o singură linie, sub forma: **expresie = rezultat** (ca în exemplu).
De exemplu, dacă fişierul **BAC.TXT** conţine: **2+3-4+6+8-5=** atunci rezultatul evaluării expresiei este **10**, afişându-se pe ecran în forma: **2+3-4+6+8-5 = 10**

36. **V_89_III_3.** Se consideră fişierul **DATE.IN** care conţine cel mult **100000** de numere naturale formate fiecare din cel mult opt cifre, separate prin câte un spaţiu. Scrieţi un program Pascal/C/C++ care scrie în fişierul **DATE.OUT** o valoare care reprezintă numărul de cifre care **NU** au apărut în niciunul din numerele aflate în fişierul **DATE.IN**.

Exemple:

DATE.IN	DATE.OUT
12 222 12 21 87 6 89 788 3 900	2

Explicaţii:
-sunt două cifre care nu au apărut în scrierea numerelor din fişierul **DATE.IN**. şi anume **4** şi **5.**

37. Fie $x_1, x_2, \ldots, x_n$ un şir format din **n** (**n≥1**) numere naturale nenule distincte. Spunem că şirul $x_1, x_2, \ldots, x_n$ este **o creastă** dacă există un indice **p** (**1<p<n**) astfel încât $x_1<x_2<\ldots<x_{p-1}<x_p$ şi $x_p>x_{p+1}>\ldots>x_{n-1}>x_n$, iar numărul aflat pe poziţia **p** (adică x_p) se numeşte **vârful crestei**.
Fişierul text **BAC.IN** conţine pe prima linie un număr natural **n** cuprins între **1** şi **100**, iar pe a doua linie conţine **n** numere naturale nenule distincte mai mici sau egale decât **5000**, despărţite prin spaţii. Scrieţi un program care să creeze un fişier text **BAC.OUT** în care să scrie, pe prima linie, numărul **0** dacă cele **n** numere aflate pe a doua linie a fişierului **BAC.IN** nu formează o creastă sau numărul care reprezintă vârful crestei, în caz contrar.

Exemple:

BAC.IN	BAC.OUT
6 7 12 17 26 15 3	26

BAC.IN	BAC.OUT
6 7 12 17 10 15 3	0

38. Fişierul text **BAC.IN** conţine, pe mai multe linii, cel mult **30000** de numere naturale nenule mai mici sau egale decât **500**, despărţite prin spaţii. Scrieţi un program care să creeze un fişier text **BAC.OUT** în care va scrie fiecare valoare distinctă din fişierul **BAC.IN**, precum şi numărul său de apariţii. Fiecare pereche cerută va fi scrisă în fişierul indicat pe câte o linie, sub forma **(valoare_distinctă,număr_apariţii)**, în ordinea crescătoare a valorilor distincte găsite. Alegeţi o metodă eficientă de rezolvare ca timp de executare. Fişierul **BAC.IN** va conţine întotdeauna cel puţin un număr.

Exemple:

BAC.IN	BAC.OUT
7 22 7 6 10 10 6 22 6 10 22 10	(6,3) (7,2) (10,4) (22,3)

39. Pe prima linie a fişierului **BAC.TXT** se găsesc numere naturale separate prin câte un spaţiu. Ele formează mai multe şiruri crescătoare de numere naturale,

sfârşitul fiecărui şir fiind marcat de valoarea **-1** (care nu face parte din nici un şir). Ştiind că valorile numerelor din şiruri nu depăşesc, în valoare absolută, **10000** şi că in total sunt cel mult **2000** de numere, scrieţi un program care să afişeze numerele comune tuturor şirurilor, în ordinea crescătoare a valorilor lor.

De exemplu, dacă fişierul conţine:

2 3 3 4 5 8 9 -1 2 4 5 8 11 -1 2 3 5 8 12 -1

atunci se va afişa: **2 5 8**

a) Descrieţi în limbaj natural o metodă eficientă de rezolvare ca timp de executare.

b) Scrieţi programul **Pascal/C/C++** corespunzător metodei descrise la punctul a).

40. Pe prima linie a fişierului **NUMERE.TXT** se află un număr natural **n** (**0<n<10000**) iar pe următoarea linie exact **n** valori naturale de cel mult **2** cifre fiecare, despărţite prin câte un spaţiu.

a) Scrieţi programul **Pascal/C/C++** care citeşte din fişier numărul **n** şi apoi cele **n** numere şi afişează pe ecran, despărţite prin câte un spaţiu numărul sau numerele care apar de cele mai multe ori pe linia a doua a fişierului. Se va utiliza un algoritm eficient (ca timp de executare şi gestionare a memoriei). De exemplu, dacă fişierul **NUMERE.TXT** are următorul conţinut:

10
21 3 4 21 4 5 21 3 6 4

atunci programul va afişa numerele **21** şi **4**, deoarece fiecare apare de câte **3** ori, iar celelalte numere apar de mai puţin de **3** ori.

b) Descrieţi pe scurt metoda folosită explicând eficienţa acesteia(3-4 rânduri)

41. Fişierul **BAC.TXT** conţine pe prima linie un număr natural cu exact **2000** de cifre din mulţimea **{0,1,...,9}**. Cifrele numărului nu sunt separate prin spaţii.

a) Scrieţi un program C/C++ care afişează pe ecran numărul cel mai mare care se poate obţine din cifrele numărului citit din fişierul **BAC.TXT**. Se va utiliza un algoritm eficient din punct de vedere al timpului de executare.

Exemplu: dacă fişierul conţine numărul **2417400...0** (**1995 de 0**) se va afişa **7442100...0** (**1995 de 0**)

b) Descrieţi metoda utilizată şi explicaţi în ce constă eficienţa ei.

42. Pe prima linie a fişierului text **NUMERE.TXT** se află o valoare naturală **n** (**0<n≤100**), iar pe următoarea linie se află exact **n** valori naturale distincte de cel mult **4** cifre fiecare, valori despărţite prin câte un spaţiu.
Scrieţi programul **Pascal** care afişează pe ecran toate numerele prime de pe a doua linie a fişierului **NUMERE.TXT**, în ordine crescătoare a valorilor lor. Numerele vor fi afişate pe o singură linie separate prin câte un spaţiu.

Dacă fişierul **NUMERE.TXT** are următorul conţinut	se afişează pe ecran
11 **2 18 31 9 4 5 7 81 22 64 3**	**2 3 5 7 31**

43. Fişierul **numere.in** conţine pe prima linie un număr natural **n** (**n<100**) iar pe a doua linie **n** numere mai mici decât **1000** separate printr-un spaţiu.
Scrieţi programul **Pascal/C/C++** ce determină câte elemente situate pe a doua linie din fişier sunt egale cu partea întreagă a mediei lor aritmetice. Exemplu: Dacă fişierul **numere.in** conţine:

5
2 3 4 3 5

se va afişa **2** (media aritmetica este **3.4** si exista **2** numere egale cu **3**)

44. Din fişierul **bac.in** se citeşte de pe prima linie un număr natural **n** (**0<n<5000**), iar de pe linia a doua se citesc **n** numere naturale, fiecare având cel mult patru cifre. În fişier numerele sunt separate prin câte un spaţiu.
a) Se cere să se aleagă un algoritm eficient pentru determinarea valorilor distincte şi să se descrie în 3-4 rânduri algoritmul justificându-se eficienţa sa;
b) Scrieţi programul **Pascal/C/C++** ce realizează prelucrarea descrisă la punctul **a**. Valorile distincte determinate se scriu, cu spaţiu între ele, în fişierul **bac.out**.

45. Prima linie a fişierului **BAC.TXT** conţine două numere naturale **m** şi **n** (**1<=n,m<=100**) iar următoarele **m** linii câte **n** numere întregi cu maxim **4** cifre fiecare, separate prin câte un spaţiu. Se cere să se afişeze pe ecran câte din cele **m** linii din fişier sunt simetrice. Spunem că o linie a fişierului este simetrică dacă elementele egal depărtate de capetele liniei respective sunt egale (primul element de pe linie este egal cu ultimul element al liniei, al doilea cu penultimul etc.). De exemplu dacă fişierul **BAC.TXT** are următorul conţinut

3 5
15 3 0 3 15
5 9 1 9 5
6 8 20 8 3

atunci se va afişa pe ecran valoarea **2** (primele două din cele **m** linii sunt simetrice).

46. Scrieţi un program Pascal care citeşte de la tastatură un număr natural nenul **k (k<100)** şi din fişierul text **BAC.TXT,** de pe prima linie un număr natural nenul **n (k<n<100.000)**, iar de pe următoarea linie un şir **S** alcătuit din **n** numere întregi formate din cel mult patru cifre fiecare, separate prin câte un spaţiu; programul va afişa pe ecran valoarea maximă care poate fi obţinută prin însumarea a **k** elemente aflate pe poziţii consecutive în şir. De exemplu, dacă se citeşte **k=5** de la tastatură, **n=10** de pe prima linie a fişierului şi numerele: **-9 11 7 -19 2 14 5 -1 6 -2** de pe a doua linie, atunci se afişează **26**, deoarece suma maximă care se poate obţine prin adunarea a **5** valori aflate pe poziţii consecutive în şir este **26** (**2+14+5+(-1)+6**).

a) Descrieţi folosind limbajul natural o metodă eficientă de rezolvare şi justificaţi eficienţa acesteia (cel mult **6** rânduri).

b) Scrieţi programul Pascal/C/C++ corespunzător metodei descrise la a).

47. Scrieţi un program Pascal care citeşte de pe prima linie a fişierului text **BAC.TXT,** cel mult **100** de numere naturale nenule formate din cel mult patru cifre fiecare, separate prin spaţii şi afişează pe ecran cifra care apare de cele mai multe ori în scrierea numerelor citite. Dacă există mai multe cifre care apar de cele mai multe ori, se vor afişa toate acestea. De exemplu, dacă din fişier se citesc numerele: **90 73 109 1248 2771**, atunci se afişează **1,7** deoarece fiecare dintre acestea apare de **3** ori.

48. În fişierul **DATE.IN** se găsesc cel mult **100000** de numere naturale de cel mult 8 cifre fiecare scrise pe o singură linie şi separate printr-un spaţiu. Scrieţi un program Pascal/C/C++ care pentu o cifră **k** citită de la tastatură, afişează pe ecran câte numere prime, în scrierea cărora s-a folosit cifra **k,** se găsesc în fişier.
Exemplu: dacă fişierul conţine numerele: 32 232 **13 31** 33 **23** 1111 19 200 5 **23** iar pentru **k** se citeşte valoarea **3,** programul va afişa **4.**

49. Fişierul **numere.in** conţine pe prima linie un număr natural **n** (**0<n<100000**), iar pe a doua linie **n** numere reale separate prin câte un spaţiu. Fiecare număr real este format din cel mult **10** cifre, inclusiv partea zecimală. Scrieţi programul **Pascal/C/C++** ce determină cifrele ce **nu** apar în scrierea nici unui număr real din fişier. Cifrele se vor afişa pe ecran în ordine crescătoare, separate prin câte un spaţiu. În cazul în care toate cifrele sunt utilizate în scrierea numerelor din fişier se va afişa pe ecran mesajul **NICI UNA**.

Exemplu: Dacă fişierul conţine pe prima linie numărul **4** şi pe a doua linie numerele: **-1.23 36 22.57 208** atunci se va afişa **4 9**.

50. În fişierul **numere.txt,** se află memorate, pe prima linie, două numere naturale, **n** şi **m** despărţite de un spaţiu (**1≤n≤10000** şi **m≤n**), iar pe

următoarea linie a fişierului, **n** numere reale despărţite prin câte un spaţiu. Scrieţi programul care citeşte din fişier datele existente şi afişează pe cea de-a treia linie a fişierului, suma obţinută din cele mai mari **m** elemente negative citite din fişier. În cazul în care fişierul nu conţine cel puţin **m** numere reale negative, se va afişa valoarea **0**.

Exemplu:

numere.txt

6 3

-14.2 -7.5 -22.33 80 1.66 -3 Suma afişată în fişier va fi **-24.7**

51. La o serbare sunt **n (0<n<21)** grupe de copii care poartă **p (0<p<11)** tipuri de uniforme(numerotate de la **1** la **p**). Scrieţi un program PASCAL/C/C++ care să afişeze pe ecran tipurile de uniforme în ordinea descrescătore a numărului total de copii ce poartă fiecare tip de uniformă. Afişarea se va face pe o singură linie, valorile fiind separate printr-un spaţiu.
Datele de intrare se citesc din fişierul text **SERBARE.TXT** care are următoarea structură: pe prima linie se află valorile lui **n**, respectiv **p**, separate printr-un spaţiu, iar pe fiecare din următoarele **n** linii câte două valori separate printr-un spaţiu ce reprezintă numărul de copii dintr-o grupă şi respectiv tipul de uniformă pe care aceştia îl poartă.

De exemplu dacă fişierul **SERBARE.TXT** are următorul conţinut:

5 3
20 2
20 3
30 2
20 1
10 1

atunci pe ecran se va afişa **2 1 3** deoarece tipul **2** de uniformă este purtat de **50** de copii, tipul **1** de **30** de copii, iar tipul **3** de **20** de copii.

52. Se citeşte un număr **n** natural, **2<n<10000** de la tastatură. Să se scrie un program PASCAL/C/C++ care creează fişierul text **BAC.TXT** şi scrie în el, pe aceeaşi linie, cu separare prin spaţiu, primele **n** cifre din şirul **1,2,3,4,5,6,7,8,9,1,2,3,...**Cifrele **i** cu **i<9** sunt consecutive, iar după fiecare cifră de **9** urmează cifra **1**.

De exemplu, dacă pentru **n** se citeşte valoarea **15**, atunci fişierul va conţine, în această ordine, valorile: **1 2 3 4 5 6 7 8 9 1 2 3 4 5** .

53. În fişierul text **MAT.TXT** se află **nxm** (**0<n<11,0<m<21**) valori numerice întregi de cel mult nouă cifre dispuse pe mai multe linii şi separate prin spaţii. Să se scrie programul **PASCAL/C/C++** care construieşte în memorie şi afişează pe ecran tabloul cu **n** linii şi **m** coloane ce se formează cu elementele din fişier astfel încât ordinea tuturor elementelor din fişier să coincidă cu parcurgerea tabloului linie cu linie.

54. Scrieţi un program Pascal/C/C++ care citeşte de la tastatură un număr natural **n (2<n<=100)** şi construieşte în memorie o matrice pătrată cu **n** linii şi **n** coloane în care prima linie conţine, în ordine, numerele **1,2,3,...,n** şi oricare altă linie este obţinută prin permutarea circulară către stânga cu o poziţie a numerelor de pe linia anterioară.

Matricea va fi afişată în fişierul text **BAC.TXT**, numerele de pe aceeaşi linie fiind separate printr-un spaţiu.

De exemplu, pentru **n=4** , conţinutul fişierului **BAC.TXT** va fi:

1 2 3 4
2 3 4 1
3 4 1 2
4 1 2 3

55. Fişierul **DATE.IN** conţine cel mult **100000** numere naturale separate prin spaţii, fiecare număr cu maximum două cifre.

Să se realizeze un program Pascal/C/C++ care scrie în fişierul **DATE.OUT** numărul numerelor din fişierul **DATE.IN** care apar exact o dată în fişier.

Exemplu: dacă fişierul **DATE.IN** conţine numerele:

45 13 **12 2 5 23 65** 13 13 13 13

fişierul **DATE.OUT** va avea următorul conţinut**: 6.**

56. Pentru o valoare **n** (număr natural, **1<=n<=100**) citită de la tastatură scrieţi programul Pascal/C care scrie în fişierul **bac.txt** un tablou bidimensional cu **n** linii şi **n** coloane cu formatul alăturat. Elementele de pe fiecare linie sunt separate prin spaţiu.

```
1  0   0   0   ...0
2  1   0   0   ...0
3  2   1   0   ...0
...................
n  n-1 n-2 n-3 ...1
```

De exemplu, pentru **n=4**, conţinutul fişierului **bac.txt** este:

```
1 0 0 0
2 1 0 0
3 2 1 0
4 3 2 1
```

57. Fişierul **matrice.txt** conţine pe primul rând două valori naturale **m** şi **n (1≤n≤100, 1≤m≤100),** reprezentând numărul de linii şi respectiv de coloane ale unei matrice **a**, iar pe următoarele **m** linii câte **n** valori întregi cu maximum **4** cifre fiecare, separate prin câte un spaţiu, reprezentând elementele matricei **a**. Se cere să se afişeze pe ecran un şir de **2*(n+m)-4** numere ordonate crescător, şir format din elementele aflate pe chenarul exterior al matricei **a**. Chenarul exterior este format din prima linie, ultima linie, prima coloană şi ultima coloană. Alegeţi un algoritm de rezolvare eficient din punct de vedere al gestionării memoriei.

De exemplu, dacă fişierul **matrice.txt** conţine:

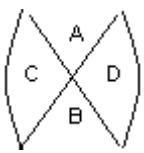
```
3 4
6 7 1 9
3 0 2 8
5 4 8 5
```

se va afişa: **1 3 4 5 5 6 7 8 8 9**

a) Descrieţi strategia de rezolvare şi justificaţi eficienţa algoritmului ales, folosind limbajul natural (**5-6** rânduri).

b) Scrieţi programul corespunzător.

58. O matrice pătrată este împărţită de cele două diagonale în patru zone notate **A, B, C, D,** conform figurii alăturate. Elementele de pe cele 2 diagonale nu aparţin nici uneia dintre cele 4 zone. Scrieţi un program **Pascal** care citeşte de la tastatură un număr



natural **n (n<=20)** şi formează în memorie o matrice pătrată cu **n** linii şi **n** coloane, în care elementele de pe cele două diagonale sunt egale cu **1**, elementele care aparţin zonelor **A** şi **B** sunt egale cu **2**, iar elementele care aparţin zonelor **C** şi **D** sunt egale cu **3**.

Elementele matricei vor fi scrise in fişierul **bac.out,** câte o linie a matricei pe câte o linie a fişierului cu spaţii între elementele fiecărei linii.

De exemplu pentru **n=5** fişierul **bac.out** va conţine:

```
1 2 2 2 1
3 1 2 1 3
3 3 1 3 3
3 1 2 1 3
1 2 2 2 1
```

59. Fişierele text **f1.txt** şi **f2.txt** conţin, fiecare, elementele a câte unui şir de numere reale **a,** respectiv **b.** Fişierele conţin maxim 100 de numere distincte scrise pe o linie şi separate printr-un singur spaţiu.

Scrieţi un program în limbajul Pascal/C/C++ care citeşte cele două şiruri de numere din fişierele **f1.txt** şi **f2.txt**, şi care scrie în fişierul **f3.txt** toate elementele comune şirurilor **a** şi **b**, pe o linie şi separate printr-un spaţiu.

Exemplu: Dacă conţinutul fişierelor **f1.txt** şi **f2.txt** este: **10.3 2.05 5 7.12**, respectiv **67 7.12 5 3 7.33 9,** atunci **f3.txt** va conţine: **7.12 5**

60. Fişierul **bac.txt** conţine pe prima linie un număr natural **N (2<N<20)**, iar pe următoarele **N** linii, câte **N** valori întregi, despărţite prin spaţii, reprezentând elementele unui tablou bidimensional, **A**. Numim **semidiagonală** a tabloului mulţimea elementelor aflate pe o direcţie paralelă cu diagonala secundară a acestuia.

Scrieţi un program Pascal care citeşte din fişierul **bac.txt** numărul **N** şi elementele tabloului **A** şi apoi citeşte de la tastatură două valori naturale **L** şi **C** (**1≤L≤N**, **1≤C≤N**)**.** Programul va afişa pe ecran suma elementelor aflate pe semidiagonala ce conţine elementul de pe linia **L** şi coloana **C**. Se va considera că liniile şi coloanele tabloului sunt numerotate începând cu **1**.

Exemplu: dacă fişierul **bac.txt** are conţinutul alăturat, iar de la tastatură se citesc **L=3** şi **C=4**, atunci se afişează pe ecran valoarea **7** (reprezentând suma elementelor evidenţiate: **7=1+2+0+4**)

```
5
1 0 2 6 4
3 2 0 5 1
1 9 4 2 3
1 2 0 4 5
5 4 3 2 1
```

61. În fişierul **MATRICE.IN** se află memorate elementele unei matrice (pe prima linie se află numerele **n** şi **m** separate printr-un spaţiu, care reprezintă numărul de linii, respectiv coloane ale matricei, iar pe următoarele **n** linii câte **m** numere, separate prin câte un spaţiu, reprezentând elementele matricei).

Să se calculeze suma elementelor matricei aflate pe rama dreptunghiulară având elementul din colţul stânga-sus determinat de indicii **i,j** iar elementul din colţul dreapta-jos este determinat de indicii **k,l**, unde **i,j,k** şi **l** sunt **4** numere naturale (**1≤i≤k≤n şi 1≤j≤l≤m**). Rama cuprinde toate elementele de pe liniile **i** şi **k**, cuprinse între coloanele **j** şi **l** inclusiv, precum şi elementele de pe coloanele **j** şi **l** cuprinse între liniile **i** şi **k** inclusiv.

Scrieţi un program **PASCAL/C/C++** care citeşte din fişier dimensiunile matricei şi elementele sale, apoi de la tastatură patru valori pentru **i,j,k,l,** calculează suma cerută şi o afişează pe ecran.

Exemplu: dacă în fişierul **MATRICE.IN** este memorată matricea din figura alăturată şi se citesc de la tastatură valorile **i=3, j=2, k=6** şi **l=6,** colţurile ramei vor avea indicii **(3,2)** şi **(6,6)**.
Se va afişa suma: **68**.

```
7  6
1  -2  3  2  1  0
2  2   5  1  0  1
9  5   2  4  2  2
5  0   5  6  0  9
11 0   1  0  5  8
12 10  9  5  5  7
5  5   2  1  8  0
```

62. Scrieţi programul Pascal/C/C++ care citeşte de la tastatură un şir de cel mult **30** de litere ale alfabetului englez şi creează fişierul text **BAC.TXT** ce conţine şirul de caractere dat şi toate prefixele acestuia de lungime cel puţin **1**, fiecare pe câte o linie, în ordinea descrescătoare a lungimii prefixelor. De exemplu, dacă se citeşte şirul: **proba**, atunci **BAC.TXT** va conţine:

```
proba
prob
pro
pr
p
```

63. Scrieţi programul Pascal/C/C++ care citeşte de la tastatură un şir de cel mult **30** de litere ale alfabetului englez şi creează fişierul text **BAC.TXT** ce conţine şirul de caractere dat şi toate sufixele acestuia de lungime cel puţin **1**, fiecare pe câte o linie, în ordinea crescătoare a lungimii sufixelor, aliniate la stânga. De exemplu, dacă se citeşte şirul **teste**, atunci **BAC.TXT** va conţine:

```
e
te
ste
este
teste
```

64. Scrieţi programul Pascal/C/C++ care citeşte de la tastatură un şir **s** de cel mult **30** de caractere şi un caracter **c**; programul determină dublarea fiecărei apariţii a caracterului **c** în **s** şi scrie noul şir obţinut în fişierul text **BAC.TXT**.

De exemplu, dacă se citeşte şirul: **alfabetar** şi caracterul **a** atunci fişierul **BAC.TXT** va conţine şirul: **aalfaabetaar**.

65. Fişierul **DATE.IN** conţine cel mult **1000** de caractere alfanumerice. Se citeşte de la tastatură un cuvânt format din cel mult **50** de caractere alfanumerice. Să se scrie un program Pascal/C/C++ care afişează pe ecran mesajul **DA** dacă toate caracterele din cuvântul citit se regăsesc în fişier, fără a conta numărul aparţiilor lor sau ordinea în care apar şi mesajul **NU** în caz contrar.
Exemplu: dacă fişierul **DATE.IN** are următorul conţinut **abc1a2bcdefgfgx7mnpo** iar cuvântul citit este **examen** răspunsul afişat pe ecran va fi **DA**

66. Fişierul text **BAC.TXT** conţine prenumele elevilor unei clase, câte un prenume (şir de maximum **20** de caractere) pe o linie a fişierului. Scrieţi un program eficient care afişează pe ecran numărul de caractere ale celui mai lung prenume precum şi numărul prenumelor având această lungime.
De exemplu, pentru fişierul alăturat valorile afişate sunt **8** **2** (**Gheorghe** şi **Vasilica** au **8** caractere).

BAC.TXT

```
Elena
Gheorghe
Andreea
Maria
Vasilica
```

67. Pe prima linie a fişierului text **BAC.TXT** se găseşte un cuvânt de exact **9** litere mici ale alfabetului englez. Pe a doua linie a fişierului se găseşte o cifră nenulă **k**. Se cere să se scrie un program care afişează pe ecran, de **k** ori, cel de al **k**-lea caracter de pe prima linie a fişierului **BAC.TXT**.

De exemplu dacă fişierul **BAC.TXT** are următorul conţinut:
algoritmi
5

se va afişa pe ecran:

rrrrr

68. Scrieţi programul **Pascal/C/C++** care citeşte din fişierul text **bac.txt** un cuvânt de cel mult **15** litere mici ale alfabetului englez şi care scrie pe ecran mesajul "**corect**" dacă în cuvânt vocalele apar în ordine alfabetică (**a, e, i, o, u**) şi niciuna dintre ele nu se repetă, iar în caz contrar scrie pe ecran mesajul "**incorect**".

De exemplu: dacă din fişierul **bac.txt** se citeşte cuvântul **tablou**, pe ecran se afişează mesajul "**corect**", dacă din fişier se citeşte cuvântul **interval** se scrie mesajul "**incorect**" pentru că vocalele nu apar în ordine alfabetică, iar dacă din acelaşi fişier se citeşte cuvântul **exemplu** se scrie mesajul "**incorect**" pentru că vocala **e** apare de două ori.

69. Se citeşte de la tastatură un număr natural **n** de cel mult **8** cifre. Să se creeze fişierul text **NR.TXT** care să conţină, câte unul pe linie, în orice ordine, toate numerele naturale distincte care se pot obţine din valoarea lui **n** prin eliminarea uneia sau mai multor cifre de la unul din capetele sale.

De exemplu, pentru **n=38604**, fişierul **NR.TXT** va conţine, câte unul pe linie şi nu neapărat în această ordine: **8604 604 4 3860 386 38 3**

70. Scrieţi un program PascalC/C++ care citeşte din fişierul text **BAC.TXT,** cel mult **100** de numere naturale aflate pe o singură linie, formate din cel mult nouă cifre fiecare, separate prin spaţii şi dintre acestea le afişează pe ecran doar pe acelea care au proprietatea de a fi palindrom. Dacă nu se citesc numere palindrom, se va afişa pe ecran valoarea **-1**. Un număr are proprietatea de a fi palindrom dacă citit de la dreapta la stânga sau de la stânga la dreapta are aceeaşi valoare. De exemplu **1221** este palindrom, în timp ce **1210** nu este palindrom.
Exemplu: dacă din fişierul **BAC.TXT** se citesc numerele: **7341 8228 660 2 80 131**, atunci pe ecran se vor afişa: **8228 2 131**

71. Fişierul **CUVINTE.IN** are pe prima linie un număr natural **n** (**0<n<1000**) iar pe următoarele **n** linii câte un cuvânt de cel mult **20** litere. Să se scrie un program care citeşte cuvintele din fişier şi scrie în fişierul **CUVINTE.OUT** cuvintele citite, în ordinea crescătoare a lungimii lor, câte un cuvânt pe o linie.

Exemplu:

CUVINTE.IN	CUVINTE.OUT
5	EXAMEN
EXAMEN	LIMBAJ
LIMBAJ	SCRISA
INFORMATICA	INFORMATICA
SCRISA	BACALAUREAT
BACALAUREAT	

72. Fişierele text **X.TXT** şi **Y.TXT** conţin fiecare numele a **7** persoane, câte un nume pe fiecare linie, fiecare nume având cel mult **15** litere. Ştiind că în fiecare fişier numele sunt memorate în ordine alfabetică, scrieţi un program Pascal/C/C++ care să citească din cele două fişiere şi să afişeze pe ecran toate numele din cele două fişiere în ordine alfabetică, separate printr-un singur spaţiu.

Ex.: Daca fişierul **X.TXT** are conţinutul: iar fişierul **Y.TXT** are conţinutul:

Ana	Angi
Dana	Cora
Daniel	Dora
Ene	Horia
Mara	Oana
Nae	Paul
Paul	Tibi

Se va afişa pe ecran :

Ana Angi Cora Dana Daniel Dora Ene Horia Mara Nae Oana Paul Paul Tibi

73. Fişierul text **cuvinte.in** conţine mai multe linii nevide de text, fiecare linie de cel mult **255** de caractere. Orice linie este compusă din unul sau mai multe cuvinte separate prin câte un spaţiu. Să se scrie un program care citeşte de la tastatură un număr natural **L** şi scrie în fişierul text **cuvinte.out** toate cuvintele de lungime **L** din fişierul de intrare, în ordinea în care apar ele în text. Cuvintele vor fi scrise pe prima linie a fişierului, separate prin câte un spaţiu.

De exemplu pentru fişierul **cuvinte.in** de mai jos şi **L=4**

A fost odată
Ca niciodată
O fata frumoasă de împărat

Fişierul **cuvinte.out** va conţine: **fost fată**

74. Fişierele text **A.TXT** şi **B.TXT** conţin fiecare vârsta a **7** persoane, câte un număr natural care reprezintă vârsta pe fiecare linie. Ştiind că în fiecare fişier vârstele sunt memorate în ordine descrescătoare, scrieţi un program Pascal/C/C++ care să citească din cele două fişiere şi să scrie pe ecran toate vârstele din cele două fişiere în ordine descrescătoare, separate printr-un singur spaţiu.

Exemplu: Dacă fişierul **A.TXT** are conţinutul: iar fişierul **B.TXT** are conţinutul:

83	79
76	75
60	72
40	45
30	30
21	25
17	18

se va afişa: **83 79 76 75 72 60 45 40 30 30 25 21 18 17**

75. Scrieţi un program Pascal/C/C++ care citeşte de la tastatură **4** şiruri de caractere formate din litere mici. Să se creeze fişierul **BAC.TXT** în care să se scrie toate perechile de şiruri dintre cele citite, perechi de forma **x,y** în care şirul **x** este subsecvenţă a şirului **y** sau şirul **y** este subsecvenţă a şirului **x**. Fiecare pereche determinată se va scrie în fişierul **BAC.TXT** pe câte un rând separate printr-o virgulă. Dacă nu există nici o astfel de pereche, în fişierul **BAC.TXT** se va scrie textul **"FĂRĂ SOLUŢIE".**

Exemplu. Se citesc: **ari**
calculator
mari
lat

Se afişează în **BAC.TXT:** **ari,mari**
calculator,lat

76. Se consideră un şir de cel mult **100** de litere mici ale alfabetului englez. Acest şir este supus unui proces de eliminare, la fiecare etapă eliminându-se toate secvenţele formate din caractere identice situate pe poziţii alăturate. Şirul nou obţinut este supus aceluiaşi proces de eliminare până când nu mai există în şir caractere alăturate egale. Scrieţi programul Pascal/C/C++ care citeşte de la tastatură şirul şi afişează în fişierul **DATE.TXT** şirul obţinut după încheierea întregului proces de eliminare.

Exemplu:
Pentru şirul **teuuueusppi** după prima aplicare a procesului de eliminare se obţine şirul **teeusi**, iar apoi şirul **tusi**. Programul va afişa în fişierul **DATE.TXT** numai şirul **tusi**, fără spaţii între literele ce-l formează.

77. Scrieţi programul **Pascal/C/C++** care citeşte de la tastatură un şir cu maximum **255** de caractere format numai din litere mici şi mari ale alfabetului englez şi îl scrie în fişierul **bac.out** "codificat" conform următoarei reguli: dacă numărul de vocale (a,e,i,o,u,A,E,I,O,U) din şir este mai mare sau egal cu numărul de consoane din şir, fiecare vocală se va înlocui cu caracterul următor din alfabetul englez, altfel fiecare consoană se va înlocui cu caracterul precedent din alfabetul englez. De exemplu, pentru şirul **bacalaureat**, fişierul va conţine şirul **bbcblbvrfbt** deoarece şirul iniţial are **6** vocale şi **5** consoane; pentru şirul **basca,** fişierul va conţine şirul **aarba** deoarece şirul iniţial are **2** vocale şi **3** consoane.

78. Fişierul **fisier.txt** conţine pe prima linie un număr natural nenul, **n**, şi pe fiecare dintre următoarele **n** linii câte un singur cuvânt format din cel mult **20** de litere mici ale alfabetului englez. Să se scrie un program care citeşte valoarea **n** şi apoi determină câte dintre cele **n** cuvinte memorate în fişier conţin secvenţa **bac**. Programul va afişa pe ecran numărul de cuvinte determinat (**0** în caz că nu există niciunul).

Exemplu: pentru fişierul alăturat se afişează **3**.

```
5
tabac
laureat
tabacar
bacterie
barca
```

79. Se consideră fişierul text **Cuvinte.in** ce conţine pe prima linie un număr natural nenul **n (n<=100)** iar pe următoarele **n** linii câte un cuvânt cu maximum 10 caractere. Să se afişeze pe ecran cuvintele din fişierul dat care sunt palindroame. În cazul în care nu există nici un cuvânt palindrom se va afişa mesajul **„NU”**. Un cuvânt este palindrom dacă citindu-l de la dreapta la stânga sau de la stânga la dreapta se obţine acelaşi cuvânt.
De exemplu dacă fişierul **Cuvinte.in** are următorul conţinut:

```
3
sas
creion
Ion
```

atunci se va afişa pe ecran: **sas**

80. Subprogramul **ordonare** primeşte prin intermediul parametrilor **c1**, **c2** şi **c3**, trei cuvinte de maximum **30** de caractere fiecare şi afişează aceste trei cuvinte în ordine lexicografică, despărţite prin câte un spaţiu.

a) Scrieţi definiţia completă a subprogramului **ordonare**.

b) Fişierul **cuvinte.in** conţine pe fiecare linie câte trei cuvinte despărţite prin câte un spaţiu. Scrieţi programul care citeşte din fişier, de pe fiecare rând al său câte trei cuvinte şi folosind apeluri ale subprogramului **ordonare** le afişează în ordine lexicografică pe câte un rând de ecran.

Exemplu:

Fişierul **cuvinte.in** conţine:	Pe ecran se va afişa:
ariadna are examen	**are ariadna examen**
azi e soare	**azi e soare**
maine va ploua	**maine ploua va**
examenul e usor	**e examenul usor**
succes elevilor participanti	**elevilor participanti succes**

81. În fişierul **puncte.txt** sunt memorate coordonatele carteziene a **n** puncte din plan. Pe prima linie a fişierului se află **n** (număr natural **n≤100**) iar apoi, pe următoarele linii ale fişierului, câte două numere naturale despărţite prin câte un spaţiu, reprezentând abscisa şi ordonata unui punct din plan. În acest plan de reprezentare se desenează pătratul determinat de punctele: **P1** de coordonate **(10,20)**, reprezentând colţul din stânga sus şi punctul **P2** de coordonate **(20,10)**, reprezentând colţul din dreapta jos al pătratului. Scrieţi programul **PASCAL/C/C++** care citeşte din fişierul **puncte.txt** datele existente şi afişează pe ecran coordonatele acelor puncte din fişier, care se află în interiorul pătratului definit de punctele **P1** şi **P2**.

Exemplu:

puncte.txt

```
4
50 60
16 18
10 12
15 11
```

Coordonatele punctelor aflate în interiorul pătratului sunt:

```
16 18
15 11
```

4. Subprograme

4.1. Subprograme nerecursive

4.1.1. Teste grilă (Pascal | C / C++)

1. Subprogramul **f** returnează prin intermediul parametrului **s** suma cifrelor numărului natural **n** transmis ca parametru. Care dintre următoarele variante este antetul corect al unui astfel de subprogram?

a.
```
procedure f(n,s:integer)
```
b.
```
procedure f(n:real; var s:integer)
```
c.
```
procedure
   f(n:integer; var s:integer)
```
d.
```
procedure f(n,s:real)
```

a.
```
void f(int n, int s)
```
b.
```
void f(float n, int &s)
```
c.
```
void f(int n, int &s)
```
d.
```
void f(float n, float s)
```

2. Parametri utilizaţi la apelul unui subprogram, se numesc:

a. parametri actuali **b.** parametri locali **c.** parametri globali **d.** parametri formali

3. Care dintre următoarele reprezintă un antet corect folosit la definirea subprogramului **P**, astfel încât programul principal alăturat să afişeze valorile **1 9**?

```
Var a,b : integer;
Begin
 a:=1; b:=5;
 P(a,b);
 write (a,' ',b)
end.
```

```
void main()
{int a=1,b=5;
 P(a,b);
 cout<<a<<' '<<b;
  /printf("%d %d",a,b);
}
```

a.
```
Procedure P(a,b:integer);
```
b.
```
Function P(a,b:integer);
```
c.
```
Function P(var x:integer; y:integer);
```
d.
```
Procedure P(x:integer; var y:integer);
```

a.
```
void P (int a,int b)
```
b.
```
int P (int a,int b)
```
c.
```
int P (int &x,int y)
```
d.
```
void P(int x,int &y)
```

4. Funcţia **f** cu parametrii **x** şi **y** determină numărul de numere întregi pare din intervalul **[x,y]**, **(x<y).** Care din următoarele expresii este adevărată pentru orice numere **x** şi **y** care **nu** au aceiaşi paritate?

a. `f(x,y)=y-x`
b. `f(x,y)=(y-x+1)DIV 2`
c. `f(x,y)=(y-x)DIV 2`
d. `f(x,y)=f(x,y+1)`

a. `f(x,y)==y-x`
b. `f(x,y)==(y-x+1)/2`
c. `f(x,y)==(y-x)/2`
d. `f(x,y)==f(x,y+1)`

5. Subprogramul **S** returnează în parametrii **d** şi **m**, cel mai mare divizor comun şi respectiv cel mai mic multiplu comun a două numere întregi transmise prin parametrii **a** şi **b**. Antetul corect al subprogramului **S** este:

Varianta Pascal	**a.**	`function S(a,b,d,m:integer):integer;`
	b.	`procedure S(a,b:integer;var d,m:integer);`
	c.	`procedure S(a,b:integer;var d,var m:integer);`
	d.	`function S(a:integer, b:integer):real;`
Varianta C/C++	**a.**	`int S(int a,int b,int d, int m)`
	b.	`void S(int a,int b,int &d,int &m)`
	c.	`void S(int a,b,int &d,&m)`
	d.	`float S(int a,b)`

6. Ştiind că funcţia **cmmdc**, cu doi parametri întregi **a** şi **b**, returnează valoarea **celui mai mare divizor comun** al numerelor **a** şi **b**, stabiliţi care dintre următoarele variante atribuie variabilei **m** valoarea **celui mai mare divizor comun** al numerelor **a,b** şi **c.**

a.	`m:=cmmdc(cmmdc(a,b),c)`	**a.**	`m=cmmdc(cmmdc(a,b),c);`
b.	`m:=cmmdc(a,b)*c`	**b.**	`m=cmmdc(a,b)*c;`
c.	`m:=cmmdc(a,b,c)`	**c.**	`m=cmmdc(a,b,c);`
d.	`m:=cmmdc(a,b) div c`	**d.**	`m=cmmdc(a,b)/c;`

7. Subprogramul **maxim(x,y)** returnează cea mai mare valoare dintre **x** şi **y**. Care dintre apelurile următoare ale funcţiei **maxim** **NU** returnează valoarea absolută a numărului real **x** ?

a.	`maxim(x,0)`	**b.**	`maxim(x,maxim(-x,0))`
c.	`maxim(x,-x)`	**d.**	`maxim(-x,maxim(x,0))`

8. Subprogramul **complement** schimbă cifrele unui număr natural **n**, astfel încât fiecare cifră **x** devine egală cu **9-x**. Care este antetul corect al unui astfel de subprogram?

a.	**procedure complement (**`var n:longint`**);**	**a.**	**void complement (**`long &n`**)**
b.	**procedure complement (**`n: real`**);**	**b.**	**void complement (**`float n`**)**
c.	**procedure complement (**`var n:float`**);**	**c.**	**void complement (**`real &n`**)**
d.	**procedure complement(**`n:longint`**);**	**d.**	**void complement(**`long n`**)**

9. Subprogramul **cifmax(i)** returnează cea mai mare cifră a numărului **i**. Stabiliţi valoarea expresiei: **cifmax(cifmax(173)+cifmax(18))**:

a. 8 **b.** 7 **c.** 15 **d.** 5

10. Subprogramul **nrcif(i)** returnează numărul de cifre pe care le are numărul natural **i**. Stabiliţi valoarea expresiei

nrcif(nrcif(173) + nrcif(18))

a. 5 **b.** 10 **c.** 2 **d.** 1

11. Subprogramul **f** are următorul antet:

```
type vector=array[0..99]of integer;
procedure f(x:vector; i:integer);
```

```
void f(int x[100], int i)
```

Care dintre următoarele apeluri este corect, ştiind că variabila **a** este declarată astfel:

```
var a:vector;
```

```
int a[100];
```

a. `f(a,a[0]);`

b. `f(a,a);`

c. `f(a[100],a[0]);`

d. `f(a[100],a);`

12. Subprogramul **verif(i)** returnează cea mai mică cifră a numărului **i** care apare de cel puţin două ori în scrierea lui **i** sau valoarea **-1** dacă numărul **i** este format din cifre distincte. Stabiliţi valoarea expresiei

verif(verif(275772) + verif(1929))

a. 1 **b.** 11 **c.** 0 **d.** 2

13. Subprogramul **nrap(s,c)** returnează numărul aparţiilor caracterului **c** în şirul **s**. Dacă **a** şi **b** sunt două şiruri de caractere formate doar din litere mici ale alfabetului englez, cu ce expresie trebuie înlocuite punctele de suspensie din secvenţa următoare astfel încât, după executarea ei, variabila de tip întreg **k** să aibă valoarea **1** dacă mulţimea formată din literele distincte ale şirului **a** este egală cu mulţimea formată din literele distincte ale şirului **b** sau valoarea **0** în caz contrar?

```
k:=1;
for c:='a' to 'z'do
   if ..... then k:=0
```

```
k=1;
for(c='a';c<='z';c++)
    if(.....) k=0;
```

Varianta Pascal

a. `(nrap(a,c)>0) and nrap(b,c)=0) or (nrap(b,c)>0 and nrap(a,c)=0)`

b. `nrap(a,c)>0 and nrap(b,c)=0`

c. `nrap(a,c)=1 and nrap(b,c)=0`

d. `(nrap(a,c)>0) and (nrap(b,c)=0) and (nrap(b,c)>0) and (nrap(a,c)=0)`

Varianta C/C++

a. `(nrap(a,c)>0 && nrap(b,c)==0) ||(nrap(b,c)>0 && nrap(a,c)==0)`

b. `nrap(a,c)>0 && nrap(b,c)==0`

c. `nrap(a,c)==1 && nrap(b,c)==0`

d. `nrap(a,c)>0 && nrap(b,c)==0 && nrap(b,c)>0 && nrap(a,c)==0`

14. Subprogramul **nrap(s,c)** returnează numărul aparițiilor caracterului **c** în șirul **s**. Știind că șirul **a** conține doar litere mici ale alfabetului englez și variabila **c** este de tip **char**, stabiliți în care dintre următoarele secvențe de program i se atribuie variabilei **k** valoarea **1** dacă și numai dacă șirul **a** este format din litere mici distincte sau valoarea **0** în caz contrar.

a.
```
k:=1;
for c:='a' to 'z'do
 if(nrap(a,c)>=2) then k:=0
```
b.
```
k:=0;
for c:='a' to 'z' do
 if(nrap(a,c)<=1) then k:=1
```
c.
```
for c:='a' to 'z' do
if(nrap(a,c)>=2)then k:=0
   else k:=1;
```
d.
```
k:=1;
for c:='a' to 'z' do
  if(nrap(a,c)=1) then k:=0
```

a.
```
k=1;
for(c='a';c<='z';c++)
  if(nrap(a,c)>=2)k=0;
```
b.
```
k=0;
for(c='a';c<='z';c++)
  if(nrap(a,c)<=1)k=1;
```
c.
```
for(c='a';c<='z';c++)
  if(nrap(a,c)>=2)k=0;
  else k=1;
```
d.
```
k=1;
for(c='a';c<='z';c++)
  if(nrap(a,c)==1)k=0;
```

15. Subprogramul **dpm(n,m)** returnează cel mai mare divizor prim al numărului natural **n**, divizor mai mic sau egal cu **m** (**2≤m≤n**). Stabiliți care dintre următoarele expresii este adevărată dacă și numai dacă numărul **n** este prim (**n≥3**):

a. `dpm(n,n)=n`
b. `dpm(n,n-1)=2`
c. `dpm(n,n)=2`
d. `dpm(n,n/2)=n div 2`

a. `dpm(n,n)==n`
b. `dpm(n,n-1)==2`
c. `dpm(n,n)==2`
d. `dpm(n,n/2)==n/2`

16. Subprogramul **invers(v,p,q)** inversează ordinea elementelor din tabloul unidimensional **v** aflate pe pozițiile **p,p+1,…,q-1,q** (se presupune că **p<=q**). Stabiliți care dintre următoarele secvențe de program inversează **doar** ordinea elementelor **v[p]** și **v[q]**:

a.
```
invers(v,p,q);
invers(v,p-1,q+1);
```
b.
```
invers(v,p,q);
invers(v,p+1,q+1);
```
c.
```
invers(v,p,q);
invers(v,p+1,q-1);
```
d.
```
invers(v,p-1,q-1);
invers(v,p,q);
```

17. Subprogramul **divmax(n)** returnează cel mai mare divizor prim al numărului **n**. De exemplu **divmax(13)** returnează valoarea **13**, iar **divmax(28)** returnează valoarea **7**.
Stabiliți valoarea expresiei: **divmax(divmax(17)+divmax(51))**;

a. 17 **b.** 68 **c.** 2 **d.** 34

18. Ştiind că iniţial, variabilele întregi x şi y au valorile **12** şi respectiv **20**, stabiliţi ce se va afişa în urma executării secvenţei de instrucţiuni

```
aplic(x,x);
write(',',x,',',y);


procedure aplic(x:byte; var y:byte);
begin
   x:=x+3;y:=y-2; write(x,',',y)
end;
```

```
aplic(x,x);
cout<<",",<<x<<",",<<y;
  /printf(",%d, %d",x,y);

void aplic(int x,int &y)
{  x+=3;y-=2;
   cout<<x<<",",<<y;
    /printf("%d,%d",x,y);
}
```

a. 13,13,13,13 **b.** 15,13,13,13 **c.** 15,10,10,20 **d.** 15,10,15,20

19. Antetul corect al unui subprogram care are ca parametri de intrare două numere întregi **a**, **b** şi returnează prin parametrii **M** şi **P**, media aritmetică şi respectiv produsul celor două numere, este :

Varianta Pascal	**a.** `function calcul(a,b:integer;var P:integer):integer;` **b.** `procedure calcul(a,b:integer;P:integer;M:real);` **c.** `procedure calcul(a,b:integer;var P:integer;var M:real);` **d.** `function calcul(a,b:integer):real;`
Varianta C/C++	**a.** `int calcul(int a, int b, int & P);` **b.** `void calcul(int a, int b, float M, int P );` **c.** `void calcul(int a, int b, float &M, int &P );` **d.** `float calcul(int a, int b);`

20. Dacă iniţial **a=3** şi **b=7**, ce se va executa în urma executării secvenţei de instrucţiuni:

```
f(b,b);
write(a,b);


procedure f(var a:integer;b:integer);
begin
 a:=a-1; b:=a+1;
 write(a,b)
end;
```

```
f(b,b);
cout<<a<<b;
  /printf("%d%d",a,b);

void f(int &a,int b)
{ a=a-1;b=a+1;
 cout<<a; /printf("d",a);
 cout<<b; /printf("d",b);
}
```

a. 6736 **b.** 6767 **c.** 6737 **d.** 2828

21. Subprogramul **putere(n)** returnează 10^{x-1}, unde **x** reprezintă numărul de cifre ale numărului **n** primit ca parametru **(n>99)**. Atunci expresia **n div putere(n)** reprezintă:

a. prima cifră a numărului **n**
b. ultima cifră a numărului **n**
c. numărul obţinut prin eliminarea ultimei cifre a numărului **n**
d. numărul obţinut prin eliminarea primei cifre a numărului **n**

22. Funcţia **numar(a)**, unde **a** este număr întreg, returnează cel mai mic întreg mai mare decât **a**, dacă **a** are cifra unităţilor egală cu cifra zecilor sau cel mai mare întreg mai mic decât **a** în caz contrar. Două valori **x** şi **y** pentru care funcţia **număr** returnează acelaşi rezultat sunt:

a. 155,156 **b.** 100,101 **c.** 12,11 **d.** 124,122

23. Subprogramul **divizori(x,y)** returnează numărul de divizori comuni a numerelor naturale **x** şi **y**. Fracţia $\frac{a}{b}$, cu **a,b** numere naturale şi $\mathbf{b} \neq \mathbf{0}$, este ireductibilă dacă este satisfăcută condiţia:

Varianta Pascal	**a.** `divizori(a,b) = 0` **b.** `divizori(a,b)/divizori(b,a)<>1` **c.** `divizori(a,divizori(b)) = 0` **d.** `divizori(a,b) = 1`
Varianta C/C++	**a.** `divizori(a,b) == 0` **b.** `divizori(a,b) / divizori(b,a) != 1` **c.** `divizori(a,divizori(b)) == 0` **d.** `divizori(a,b) == 1`

24. Subprogramul **max(n)** returnează cea mai mare cifră a numărului natural **n**. Pentru **n** număr natural, format din **3** cifre, expresia

Varianta Pascal	`max(n div 10 mod 10)+max(n mod 10)+ max(n div 100)`
Varianta C/C++	`max(n/10%10)+max(n%10)+max(n/100)`

reprezintă:

a. Cifra unităţilor numărului **n** **b.** Cifra maximă a numărului **n**

c. Numărul cifrelor numărului **n** **d.** Suma cifrelor numărului **n**

25. Care succesiune de instrucţiuni este corectă, ştiind că se foloseşte subprogramul Pascal cu antetul **procedure f(a:integer,var b:integer),** respectiv C/C++: **void f(int a, int &b)**?

a.
```
a:=1; b:=1; f(a,b)
```
b.
```
b:=1; f(b,1)
```
c.
```
a:=1; f(a,1)
```
d.
```
a:=1;b:=1;write(f(a,b))
```

a.
```
a=1; b=1; f(a,b);
```
b.
```
b=1; f(b,1);
```
c.
```
a=1; f(a,1);
```
d.
```
a=1; b=1; cout<<f(a,b);
     / printf("%d",f(a,b));
```

26. Fie subprogramul **f** definit alăturat şi variabilele **a** şi **b** de tip **integer/int** , cu valorile **a=1** şi **b=2**. Ce valori vor avea variabilele **a** şi **b** în urma apelului **f(a,b+1)**?

```
procedure f(var x:integer;y:integer);
begin
 x:=x+y;  y:=x-y; x:=x-y
end;
```

```
void f(int &x,int y)
{ x=x+y;
  y=x-y;  x=x-y;
}
```

a. a=2,b=1 **b.** a=3,b=2 **c.** a=1,b=2 **d.** a=2,b=2

27. Pentru a calcula aria unui triunghi cu lungimile laturilor memorate de variabilele reale **a, b** şi **c**, se utilizează funcţia **arie** cu definiţia următoare.

```
function arie(a,b,c:real):real;
var p:real;
begin
p:=(a+b+c)/2;
arie:=sqrt(p*(p-a)*(p-b)*(p-c))
end;
```

```
float arie(float a,float b, float c)
{ float p;
  p=(a+b+c)/2;
  return sqrt(p*(p-a)*(p-b)*(p-c));
}
```

Care dintre următoarele instrucţiuni **nu** este corectă?

a.
```
a:=arie(a,b,c);
```
b.
```
write(arie(a,b,c));
```
c.
```
if (arie(a,b,c)>10)then
          a:=arie(a,b,c);
```
d.
```
arie:=arie(a,b,c);
```

a.
```
a=arie(a,b,c);
```
b.
```
printf("%f ",arie(a,b,c));
/ cout<<arie(a,b,c);
```
c.
```
if (arie(a,b,c)>10)
      a=arie(a,b,c);
```
d.
```
arie=arie(a,b,c);
```

28. Fie funcţia **f** definită mai jos:

```
function
f(var a,b:integer):integer;
begin
 while (a <>b) do
    if (a>b) then a:=a-b
             else b:=b-a;
   f:=a
end;
```

```
int f(int &a, int &b)
{ while (a !=b)
   if (a>b) a=a-b;
       else b=b-a;
  return a;}
```

Ce se afişează ca urmare a executării secvenţei următoare?

```
a:=4; b:=18;
write(f(a,b));write(a);
write(b)
```

```
a=4; b=18;
printf("%d",f(a,b));  /cout<<f(a,b);
printf("%d %d",a,b);  /cout<<a<<b;
```

a. 2 2 2 **b.** 2 4 4 **c.** 4 18 2 **d.** 2 4 18

29. Subprogramul **p** realizează permutarea circulară a numerelor reţinute de parametri de tip int **x,y şi z** cu o poziţie spre dreapta, noile valori fiind trasmise tot prin intermediul parametrilor. De exemplu pentru **x=1, y=2, z=3,** în urma apelului **p(x,y,z)** obţinem **x=3, y=1, z=2**. Dacă valorile iniţiale ale variabilelor de tip int x, y şi z sunt : **x=1, y=2, z=3**, atunci după executarea apelurilor succesive : **p(y,z,x); p(z,y,x);** noile valori memorate de variabilele **x, y, z** vor fi:

a. 1,2,3 **b.** 3,1,2 **c.** 1,3,2 **d.** 3,2,1

30. Subprogramul **med(a,b)** returnează media aritmetică a numerelor reale primite prin intermediul parametrilor **a** şi **b**. Pentru **x,y,z** şi **t,** variabile de tip real, selectaţi instrucţiunea care atribuie variabilei **x** suma dintre media aritmetică a valorilor **y** şi **z** şi media aritmetică a valorilor **z** şi **t**.

a. `x:=(y+z+t)/2;`	**a.** `x=(y+z+t)/2;`
b. `x:=med(med(y,z),t);`	**b.** `x=med(med(y,z),t);`
c. `x:=med(z,y)+med(y,t);`	**c.** `x=med(z,y)+med(y,t);`
d. `x:=med(t,y)+z;`	**d.** `x=med(t,y)+z;`

31. Funcţia următoare:

```
function max(a,b:integer):integer;
```
```
int max(int a, int b)
```

returnează maximul dintre valorile variabilelor **a** şi **b**. Cu ce instrucţiune se înlocuiesc punctele de suspensie astfel încât la finalul executării secvenţei alăturate să se afişeze maximul dintre elementele vectorului **v**, care are **20** de elemente întregi?

```
t:=max(v[1],v[2]);
for i:=3 to 20 do
      ..............  ;
writeln(t);
```

a. `t:=max(v[i],v[i+1]);`

b. `t:=max(max(v[i],v[1]),v[i+1]);`

c. `t:=max(t,v[i]);`

d. `t:=max(v[1],v[i]);`

```
t=max(v[0],v[1]);
for(i=2;i<20;i++)
        ..............  ;
cout<<t; / printf("%d",t);
```

a. `t=max(v[i],v[i+1]);`

b. `t=max(max(v[i],v[0]),v[i+1]);`

c. `t=max(t,v[i]);`

d. `t=max(v[0],v[i]);`

32. Subprogramul **Sub** are următorul antet:

```
function Sub( v: vector;
          n, k: integer):real;
```
```
float Sub(float v[50],int n,int k)
```

Considerând următoarele declarări:

```
var x:vector;
    y,z:real; a,b,c:integer;
```
```
float x[50],y,z; int a,b,c;
```

care variantă reprezinta o apelare corectă a subprogramului **Sub**?

a. `z:= Sub(x,a,b);`
b. `x:= Sub(x,y,z);`
c. `x:= Sub(x,a,b);`
d. `y:= Sub(a,b,c);`

a. `z=Sub(x,a,b);`
b. `x=Sub(x,y,z);`
c. `x=Sub(x,a,b);`
d. `y=Sub(a,b,c);`

33. În subprogramul alăturat, **y** este:

```
function f(x:real):integer;
var y:integer;
begin
 if x>0 then y:=1
        else y:=-1;
 f:=y
end;
```

```
int f(float x)
{int y;
  if(x>0)y=1;
         else y=-1;
  return y;
}
```

a. variabilă locală
b. variabilă globală
c. parametru formal
d. numele funcţiei

34. Ce se va afişa în urma executării programului alăturat?

```
procedure F(var a:integer;
 b: integer);
begin
 b:= b + a; a:= a * b;
 Write(a, ' ', b, ' ')
end;
var x, y : integer;
begin
       x := 5; y := 7;
       F(x, y);
      write(x, ' ', y)
end.
```

```
#include <stdio.h>
/ #include <iostream.h>
void F(int &a, int b)
{ b = b + a; a = a * b;
  printf("%d %d ",a,b);
    /  cout<<a<<" "<<b<<" ";
}
void main()
{ int x = 5, y = 7;
  F(x, y);
  printf("%d %d", x, y);
    /  cout<<x<<" "<<y;
}
```

a. 12 60 5 7
b. 60 12 5 7
c. 60 12 60 7
d. 60 12 60 12

35. Se consideră subprogramul **f** al cărui antet este:

```
Procedure f(var a:matrice;
            n,m,i,j:integer);
```

```
void f(int a[100][100],
       int n,int m,int i,int j);
```

Subprogramul realizează interschimbarea elementelor liniilor **i** şi **j** ale tabloului transmis prin parametrul **a**, care are **n** linii şi **m** coloane. Pentru a ordona crescător numerele de pe coloana **a 3-a** a tabloului **a**, functia **f** se apelează în timpul executării secvenţei alăturate de instrucţiuni:

```
for i:=1 to n-1 do
 for j:=i+1 to n do
  if(a[i,3]>a[j,3])
    f(a,n,m,i,j);
```

```
for(i=0;i<n-1;i++)
 for(j=i+1;j<n;j++)
  if(a[i][3]>a[j][3])
    f(a,n,m,i,j);
```

a. de **m** ori

b. de **n** ori

c. **o** dată

d. de cel mult $\frac{n(n-1)}{2}$ ori

4.1.2. Probleme (limbaj pseudocod, Pascal | C/C++)

1. Se consideră programul pseudocod alăturat, în care **mult(n,a)** semnifică puterea maximă la care apare numărul natural **a** în descopunerea în factori a lui **n**, iar **max(x,y)** semnifică maximul numerelor naturale **x** şi **y**.

```
citeşte m,n
            (m,n∈N, prime între ele)
z←max(mult(n,2),mult(n,5))
r←10*m
┌pentru i=1,z execută
│   r←r%n*10
└■
┌dacă r≠0 atunci scrie „D"
│          altfel scrie „N"
└■
```

1) Care este valoarea afişată dacă se citesc valorile **m=10** şi **n=3**?

2) Dacă **m=9**, precizaţi o valoare pentru variabila **n** astfel încât să se afişeze "**N**".

3) Dacă **m=10**, atunci câte **cifre** pot fi introduse ca valoare pentru **n** astfel încât să se afişeze "**D**"?

4) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

2. se consideră programul pseudocod alăturat:

Se consideră funcţia **max(x,y)** care calculează maximul dintre numere întregi **x** şi **y**.

```
citeşte n
     (număr natural nenul)
┌pentru i=1,n execută
│   citeşte V_i
└■
m ← V_1; i ← 3
┌cât timp i<= n execută
│  m ← max(m, V_i)
│  i ← i+2
└■
scrie m
```

1) Ce se va afişa pentru **n=5** şi V_1**=4,** V_2**=-8,** V_3**=5,** V_4**=10,** V_5**=2.**

2) Rescrieţi algoritmul modificând un număr minim de linii din algoritmul iniţial astfel încât acesta să afişeze valoarea maximă din vectorul **V**.

3) Scrieţi în limbajul Pascal/C/C++ definiţia completă a funcţiei **max(x,y)**.

4) Scrieţi programul Pascal/C/C++ corespunzător cu algoritmul dat.

3. Se consideră programul pseudocod alăturat:

Se consideră funcţia **f(n)** care calculează suma primelor **n** numere naturale.

(**f(n)=1+2+3+...+n**)

```
citeşte k,n
(numere naturale nenule, k<n)
p←1
┌pentru i=k,n execută
│   p←p * f(i)
└■
scrie p
```

1) Ce se va afişa pentru **k=3** şi **n=5**?

2) Daţi un exemplu de valori pentru **k** şi **n** astfel încât să se afişeze **3150**.

3) Scrieţi în limbajul Pascal/C/C++ definiţia completă a funcţiei **f(n)**.

4) Scrieţi programul Pascal/C/C++ corespunzător cu algoritmul dat.

4. Se consideră programul pseudocod alăturat:

Se consideră subprogramul **interschimb(x,y)** care realizează interschimbarea valorilor variabilelor întregi **x** şi **y** (dacă **x=2** şi **y=7**, după apelul **interschimb(x,y)** rezultă **x=7** şi **y=2**)

```
citeşte a,b,c (numere întregi)
┌dacă a>b atunci
│     interschimb(a,b)
└■
┌dacă a>c atunci
│     interschimb(a,c)
└■
┌dacă b>c atunci
│     interschimb(b,c)
└■
scrie a, b, c
```

1) Ce se va afişa pentru **a=10**, **b=3**, **c=6**?

2) Ce relaţie matematică trebuie să existe între valorile citite pentru **a**, **b**, **c** astfel încât programul pseudocod să nu efectueze niciun apel al subprogramului **interschimb**?

3) Scrieţi în limbajul Pascal/C/C++ definiţia completă a subprogramului **interschimb(x,y)**.

4) Scrieţi programul Pascal/C/C++ corespunzător algoritmului dat.

5. Se spune despre un număr natural că este superprim dacă atât el cât şi toate prefixele lui sunt numere prime. De exemplu, numărul **313** este un număr superprim pentru că **313** este prim şi prefixele: **3, 31** sunt numere prime.
Scrieţi un program Pascal/C/C++ care citeşte de la tastatură un număr natural **n** cu cel mult opt cifre şi care verifică dacă **n** este superprim. Programul afişează pe ecran mesajul **DA** dacă numărul este superprim, altfel va afişa mesajul **NU**.

6. Realizaţi următoarele cerinţe utilizând limbajul Pascal/C/C++:

a) Scrieţi definiţia completă a unui subprogram **sub** cu doi parametri care:

- primeşte prin intermediul parametrilor **a** şi **b** două numere naturale formate fiecare din cel mult **8** cifre;
- elimină cifra unităţilor numărului conţinut de parametrul **a**, mutând-o la sfârşitul numărului conţinut de parametrul **b**;
- returnează prin intermediul parametrilor **a** şi **b** noile numere obţinute.

De exemplu, pentru valorile **1234** şi **56** ale parametrilor **a** şi **b**, în urma executării subprogramului **sub** valorile returnate prin intermediul parametrilor vor fi **123** şi **564**.

b) Scrieţi programul care citeşte de la tastatură un număr natural **n** cu cel mult **8** cifre şi care verifică dacă numărul **n** este un palindrom, folosind apeluri ale subprogramului **sub**. Programul va afişa pe ecran mesajul **DA** dacă numărul este palindrom, altfel va afişa mesajul **NU**. (Un număr natural **n** este palindrom dacă este egal cu numărul obţinut prin scrierea cifrelor numărului **n** în ordine inversă.)

De exemplu, dacă **n=12321** atunci se va afişa pe ecran mesajul **DA**, iar dacă **n=124321** atunci se va afişa pe ecran mesajul **NU**.

7. Realizaţi următoarele cerinţe utilizând limbajul Pascal/C/C++:

a) Scrieţi definiţia completă a unei funcţii **nr** cu doi parametri, funcţie care:

- primeşte prin intermediul parametrilor **a** şi **b** două numere reale pozitive;
- returnează numărul de numere întregi cuprinse între valorile parametrilor **a** şi **b**, inclusiv.

De exemplu, pentru valorile **10.5** şi **7** ale parametrilor **a** şi **b**, funcţia va returna valoarea **4** deoarece între valorile **7** şi **10.5** sunt **4** numere întregi: **7**, **8, 9** şi **10**.

b) Scrieţi programul în care se citesc de la tastatură numerele naturale nenule **a** şi **k** (**a∈[1000,9999], 4 ≤ k ≤ 9**) şi în care se determină câte numere naturale de câte **k** cifre se divid cu **a,** folosind apeluri ale subprogramului **nr**. Programul va afişa pe ecran numărul de numere determinat.

De exemplu, dacă **a=2007** şi **k=4**, atunci numărul afişat este **4**, deoarece sunt **4** de numere de **4** cifre care se divid cu **2007** (**2007, 4014, 6021, 8028**).

8. Ştiind că sunt definite subprogramele:

- **S1** care primeşte prin intermediul parametrului **n** un număr natural de cel mult **8** cifre şi care returnează numărul de cifre ale numărului **n**;
- **S2** cu doi parametri, care primeşte prin intermediul parametrului **n** un număr natural de cel mult **4** cifre şi prin intermediul parametrului **k** o cifră. Subprogramul dublează fiecare apariţie a cifrei **k** în numărul **n** şi returnează numărul nou format prin intermediul parametrului **n**.

(De exemplu, pentru **n=1232** şi **k=2** valoarea returnată este **122322,** iar pentru **n=1232** şi **k=4** valoarea returnată este **1232**.)

a) Scrieţi numai antetul fiecăruia dintre cele două subprograme.

b) Scrieţi în limbajul Pascal/C/C++ programul în care se citesc de la tastatură un număr natural **n** de cel mult **4** cifre şi două cifre distincte **a** şi **b.** Programul va determina şi va afişa câte dintre cifrele numărului **n** sunt diferite atât de **a** cât şi de **b**, fără a accesa cifrele numărului **n**, folosind apeluri ale subprogramelor **S1** şi **S2**. Programul va afişa pe ecran numărul determinat.

De exemplu, dacă **n=1215**, **a=3** şi **b=2**, atunci numărul determinat este **3**, deoarece **n** are trei cifre diferite de **3** şi de **2.**

9. Realizaţi următoarele cerinţe utilizând limbajul Pascal/C/C++:

a) Scrieţi definiţia completă a unei funcţii **produs** cu un parametru, funcţie care:

- primeşte prin intermediul parametrului **a** un număr natural cu cel mult **8** cifre (**a>1**);
- returnează produsul divizorilor primi ai valorii parametrului **a**.

De exemplu, pentru valoarea **300** a parametrului **a**, funcţia va returna valoarea **30** deoarece **a=2^2*3*5^2** şi **2*3*5=30**.

b) Scrieţi programul în care se citesc de la tastatură numerele naturale nenule **a,b** şi **c** de cel mult **8** cifre fiecare (**a>1, b>1, c>1**) şi în care se verifică dacă numerele citite au exact aceiaşi factori în descompunerea lor în factori primi, folosind apeluri ale subprogramului **produs**. Dacă cele trei numere au exact aceiaşi factori (exponenţii lor putând să difere), programul va afişa pe ecran mesajul **DA**, în caz contrar va afişa mesajul **NU**.

De exemplu**,** dacă **a=300**, **b=1500** şi **c=30**, atunci se va afişa mesajul **DA**, iar pentru numerele **a=300**, **b=700** şi **c=140**, se va afişa mesajul **NU**.

10. Se defineşte subprogramul **sub** cu doi parametri, subprogram care primeşte prin intermediul parametrului **n** un număr natural de cel mult opt cifre şi prin intermediul parametrului **k** o cifră. Subprogramul determină eliminarea primei apariţii a cifrei **k** în numărul **n** şi returnează numărul obţinut prin intermediul parametrului **n**. De exemplu, pentru valorile **n=152422** şi **k=2** ale parametrilor, în urma apelului subprogramului **sub**, parametrul **n** va returna valoarea **15422**. Dacă cifra **k** nu apare în scrierea numărului **n**, atunci valoarea lui **n** nu se modifică.

a) Scrieţi definiţia completă a subprogramului **sub.**

b) Scrieţi în limbajul Pascal/C/C++ programul în care se citesc de la tastatură două numere naturale **a** şi **b** cu cel mult opt cifre. Programul va determina şi va afişa pe ecran numărul de cifre distincte ce intră atât în scrierea lui **a** cât şi în scrierea lui **b**, fără a accesa cifrele numerelor **a** şi **b**, folosind apeluri ale subprogramului **sub**.

De exemplu, pentru valorile **a=1237248** şi **b=1245823** programul va determina afişarea pe ecran a valorii **5** deoarece sunt cinci cifre distincte (**1,2,3,4** şi **8**) ce apar atât în scrierea lui **a** cât şi în scrierea lui **b**.

11. Se consideră subprogramele **putere** şi **maxim**. Subprogramul putere primeşte ca parametru un număr natural **n** (**n≤9**) şi returnează valoarea 10^n. Subprogramul **maxim** primeşte prin intermediul parametrului **n** un număr natural cu cel mult 9 cifre şi returnează prin intermediul parametrului **max** cifra sa maximă, iar prin intermediul parametrului **k**, poziţia primei apariţii a acesteia în numărul **n** (numărând de la dreapta spre stânga, cifra unităţilor fiind pe poziţia **0**). Exemplu: dacă **n=328785** cifra sa maximă este **8**, iar poziţia primei apariţii este **1**.

a) Scrieţi definiţia completă a subprogramului **maxim** şi numai antetul subprogramului **putere**.

b) Scrieţi un program care citeşte de la tastatură un număr natural **n** cu cel mult 9 cifre toate nenule şi, apelând subprogramele **putere** şi **maxim**, construiţi cel mai mare număr care se poate forma din cifrele numărului **n** şi afişaţi rezultatul.

Exemplu: dacă **n=4898721** se va afişa **9887421**.

12. a) Scrieţi în limbajul Pascal/C/C++ definiţia completă a subprogramului **f** care primeşte prin intermediul parametrului **n** un număr natural de cel mult **8** cifre iar prin intermediul parametrului **k** un număr prim cu cel mult **8** cifre. Subprogramul returnează prin parametrul **p** numărul care reprezintă **puterea** la care apare **k** în descompunerea în factori primi a numărului **n** şi prin parametrul **n** **câtul** obţinut prin împărţirea numărului **n** la numărul k^p.

Exemplu. Dacă **n** şi **p** sunt variabile întregi iar **n** reţine iniţial valoarea **500**, în urma apelului **f(n,5,p)** **n** va primi valoarea **4** iar **p** va primi valoarea **3**.

b) Scrieţi un program Pascal/C/C++ care citeşte de la tastură un număr natural nenul **n** cu cel mult opt cifre şi care folosind apeluri ale subprogramului definit la punctul a) determină o valoare minimă **b** care verifică relaţia

$$n = a^2 \cdot b, \qquad a,b \in N^*$$

Exemple: pentru **n=21560** se afişează **110** deoarece $21560=14^2 \cdot 110$; există şi alte posibilităţi de a-l scrie pe **21560** sub forma cerută, dar în acestea valoarea lui **b** este mai mare decât **110** (de exemplu, $21560 = 7^2 \cdot 440$)

13. **a)** Scrieţi definiţia completă a unui subprogram care primeşte prin parametru **n** un număr natural par **(4≤n≤2.000.000.000)** şi care returnează prin intermediul parametrilor **p1** şi **p2** **cel mai mare număr prim mai mic decât n şi respectiv cel mai mic număr prim mai mare decât n**. De exemplu, pentru **n=16** subprogramul va returna numerele **13** şi **17**.

b) Realizaţi un program în limbajul Pascal/C/C++ care pentru un număr natural **k (4≤k≤200)** citit de la tastatură afişează pe ecran, separate printr-un

spaţiu, cele mai mici **k** numere naturale pare care au proprietatea că sunt încadrate de două numere prime; spunem că un număr natural **p** este încadrat de două numere prime dacă numerele **p-1** şi **p+1** sunt ambele prime. Programul va apela în mod util subprogramul definit la punctul a).

Exemplu: pentru **k=4** se vor afişa **:4 6 12 18**

14. Se consideră subprogramul **NRDIV**, cu **3** parametri, care:
- primeşte prin intermediul parametrului **n** un număr natural nenul, **2≤n≤10000**;
- returnează prin intermediul parametrilor **d1** şi **d2** cel mai mic, respectiv cel mai mare divizor prim al său (**1<d1≤d2≤n**).

Pentru **n=6**, se obţin **d1=2** şi **d2=3**, pentru **n=8**, se obţin **d1=2** şi **d2=2**, iar pentru **n=7** se obţin **d1=7** şi **d2=7**.

a) Scrieţi definiţia completă a subprogramului **NRDIV**

b) Scrieţi un program C/C++ care citeşte două numere naturale **a** şi **b,** de cel mult **4** cifre fiecare, şi determină prin apeluri utile ale subprogramului **NRDIV** toate numerele naturale cuprinse în intervalul **[a,b]** cu proprietatea că toţi divizorii lor primi au exact două cifre. Numerele determinate vor fi afişate pe ecran, cu spaţii între ele.

Exemplu: dacă **a=120, b=200** se vor afişa numerele: **121 143 169 187**

15. **a)** Scrieţi definiţia completă a subprogramului **ncif** care primeşte un număr natural de cel mult 9 cifre prin intermediul parametrului **n** şi returnează prin intermediul parametrului **c** numărul de cifre ale lui **n**.

b) Scrieţi programul Pascal/C/C++ care rezolvă următoarea problemă:

În fişierul text **DATE.TXT** se află **100000** de valori naturale de cel mult 8 cifre fiecare; valorile sunt separate prin câte un spaţiu. Să se afişeze pe ecran câte valori din fişier au un număr maxim de cifre. Se vor folosi apeluri ale subprogramului **ncif**. Alegeţi o metodă eficientă din punctul de vedere al gestionării memoriei.

16. Subprogramul **panta** primeşte prin intermediul singurului parametru **n** un număr natural de cel mult **9** cifre şi returnează **diferenţa** dintre cea mai mare cifră şi cea mai mică cifră a numărului **n** .

De exemplu, pentru **n=23498** subprogramul returnează valoarea **7**, iar pentru **n=222** subprogramul returnează **0**.

a) Scrieţi definiţia completă a subprogramului **panta**.

b) Se citeşte de la tastatură un număr natural **k** format din cel mult **9** cifre **distincte**.

Scrieţi declarările de date şi programul principal în care se verifică, folosind apeluri ale subprogramului **panta**, dacă numărul natural **k**, este format din cifre consecutive, aflate în orice ordine. Programul va afişa pe ecran mesajul **DA** în caz afirmativ şi mesajul **NU** altfel. De exemplu, pentru **k=25436** se va afişa **DA** (fiind format din cifrele consecutive **2,3,4,5,6**), iar pentru **k=2364** se va afişa **NU** (lipsind cifra **5**).

17. a) Scrieţi definiţia completă pentru un subprogram care primeşte prin singurul său parametru un număr natural nenul de maximum **4** cifre şi care returnează valoarea **1** dacă numărul transmis ca parametru are un număr impar de divizori şi **0** dacă acesta are un număr par de divizori. Alegeţi o metodă eficientă de rezolvare.

b) Pe prima linie a fişierului text **BAC.TXT** se află un număr natural **n** de cel mult **6** cifre; a doua linie a fişierului conţine **n** valori naturale de cel mult **3** cifre fiecare, cu spaţii între ele. Scrieţi un program C/C++ care determină valorile de pe a doua linie a fişierului care au un număr impar de divizori şi afişează, cu exact două zecimale, media aritmetică a valorilor determinate. Dacă nu există nicio valoare cu proprietatea cerută, se va afişa mesajul **NU EXISTA**. În program se vor folosi în mod util apeluri ale subprogramului definit la a).

De exemplu, dacă fişierul **BAC.TXT** are următorul conţinut:

```
6                          prima linie
1 10 3 14 196 90           a doua linie
```

programul va afişa valoarea **98.50** deoarece pe a doua linie a fişierului se află două numere care au un număr impar de divizori şi anume **1** şi **196,** iar media lor aritmetică este **98.5**.

18. a) Scrieţi definiţia completă pentru un subprogram cu trei parametri:
- **x** , o matrice pătrată cu elemente reale, având maximum **10** linii numerotate de la **0** la **9** şi **10** coloane numerotate de la **0** la **9**;
- **n** , număr natural ce reprezintă numărul efectiv de linii şi de coloane ale matricei **x**;
- **p** , număr natural, **0≤p≤n-1**;

Subprogramul va returna suma elementelor aflate pe linia **p** a matricei **x**.
b) Să se scrie un program care rezolvă următoarele cerinţe:
- citeşte de la tastatură o valoare naturală reprezentând numărul de linii şi numărul de coloane ale unei matrice pătratice, apoi citeşte tot de la tastatură elementele reale ale acestei matrice;
- utilizând apeluri ale subprogramului definit la punctul a), scrie în fişierul text **BAC.OUT** elementele din matrice aflate pe linia care are proprietatea că suma elementelor de pe respectiva linie este maximă. În fişierul text valorile vor fi scrise pe o singură linie, separate printr-un spaţiu. Dacă există mai multe linii care au aceeaşi sumă maximă, se vor scrie elementele uneia dintre ele.

19. a) Scrieţi definiţia completă pentru un subprogram cu un parametru **n**, număr natural nenul cu cel mult **6** cifre, care returnează valoarea expresiei:

$$1+\frac{1}{\sqrt{2}}+\frac{1}{\sqrt{3}}+\frac{1}{\sqrt{4}}+\ldots+\frac{1}{\sqrt{n}}$$

b) Să se scrie un program care, utilizând apeluri ale subprogramului definit la cerinţa a), rezolvă următoarea problemă: se citeşte de pe prima linie a fişierului **bac.in** o valoare naturală nenulă **n**, **(1≤n≤100000, n impar)**; să se scrie în fişierul **bac.out**, cu exact trei zecimale, valoarea expresiei:

$$1-\frac{1}{\sqrt{2}}+\frac{1}{\sqrt{3}}-\frac{1}{\sqrt{4}}+\ldots-\frac{1}{\sqrt{n-1}}+\frac{1}{\sqrt{n}}$$

Exemplu:

bac.in	**bac.out**
5	**0,817**

20. Funcţia **count** are trei parametri:
- **x** un vector de 2500 de elemente reale **distincte sortate crescător**;
- **a** şi **b** doi parametri reali;

Funcţia returnează numărul de elemente din vectorul **x** care se află în intervalul închis determinat de valorile **a** şi **b** (**[a,b]** dacă **a<b** sau **[b,a]** dacă **b≤a**).

a.) scrieţi definiţia completă a funcţiei **count** implementând un algoritm eficient din punct de vedere al timpului de executare pentru calculul valorii pe care funcţia trebuie să o returneze;

b.) faceţi o descriere în limbaj natural a algoritmului implementat la cerinţa a.) explicând în ce constă eficienţa acestui algoritm.

21. Subprogramul **interval** primeşte prin parametru **x** o valoare reală iar prin parametri **a** şi **b** două valori întregi (**-9999≤a,b≤9999**). Subprogramul returnează valoarea **1** dacă **x** aparţine intervalului închis determinat de **a** şi **b** şi **0** în caz contrar. Intervalul închis determinat de **a** şi **b** este **[a,b]** dacă **a<b** sau **[b,a]** dacă **b≤a.**

a) Scrieţi definiţia completă a subprogramului **interval**;

b) Fişierul **numere.in** conţine pe prima linie, separate printr-un spaţiu, numere reale. Scrieţi un program care citeşte de la tastatură două valori întregi **a** şi **b,** fiecare dintre ele având cel mult patru cifre şi scrie în fişierul **numere.out** toate numerele din fişierul **numere.in** care aparţin intervalului închis determinat de valorile **a** şi **b**. În fişierul **numere.out** valorile vor fi scrise cu exact două zecimale, cu spaţii între ele.

Exemplu: dacă fişierul **numere.in** conţine valorile

3.41532 -10.1217 8 1234.5678 1.9111 -2.3108

şi de la tastatură se citesc pentru **a** şi **b** valorile **10** şi respectiv **-3** atunci fişierul **numere.out** va avea următorul conţinut:

3.41 8.00 1.91 -2.31

sau

3.42 8.00 1.91 -2.31

22. Scrieţi un subprogram **vocal** care are ca parametru un şir de caractere **s** format din cel mult **50** de litere ale alfabetului englez şi returnează numărul de vocale (**a,e,i,o,u**) din şirul respectiv.

23. Scrieţi definiţia completă a unui subprogram **matdiag** care primeşte prin intermediul primului parametru, **n**, un număr natural nenul mai mic sau egal decât **20** şi prin intermediul celui de-al doilea parametru, **a**, un tablou bidimensional cu **n** linii şi **n** coloane, format din numere întregi din intervalul

[-100,100] şi care returnează valoarea **1** dacă pe fiecare linie elementul corespunzător diagonalei principale este mai mare sau egal cu suma celorlalte elemente de pe linia respectivă sau returnează valoarea **0** în caz contrar**.**

24. Scrieţi definiţia completă a unui subprogram **vect01** care primeşte prin intermediul primului parametru, **n**, un număr natural nenul mai mic sau egal cu **20,** prin intermediul celui de-al doilea parametru, **v**, un tablou unidimensional cu **n** elemente format din numere întregi din intervalul **[-100,100]** şi care returnează valoarea **1** dacă tabloul este format din **n-1** valori egale cu **0** şi o singură valoare egală cu **1**; în orice alt caz subprogramul returnează valoarea **0**.

25. Considerăm definit un subprogram **nrap** care are doi parametri: un şir de caractere **s** de lungime cel mult **100** şi un caracter **c**. Subprogramul **nrap** întoarce numărul aparţiţilor caracterului **c** în şirul **s**.
a) Scrieţi numai antetul subprogramului **nrap**.
b) Scrieţi un program care citeşte de la tastatură două şiruri de caractere formate fiecare din cel mult **100** de litere ale alfabetului englez şi, apelând subprogramul **nrap** descris mai sus, verifică dacă cele două şiruri sunt formate din exact aceleaşi caractere, eventual în altă ordine. În caz afirmativ programul va afişa pe ecran mesajul **"DA"**, altfel va afişa **"NU"**.

26. Scrieţi definiţia completă a unui subprogram **sumazec** cu trei parametri care primeşte prin intermediul primului parametru **a** un număr natural nenul având maxim 8 cifre, prin intermediul celui de-al doilea parametru **n** un număr natural nenul format dintr-o singură cifră şi returnează prin intermediul celui de-al treilea parametru **s** suma primelor **n** zecimale ale numărului real $\mathbf{a}^{-1}$. De exemplu, **sumazec(4,3,s)** va returna prin intermediul parametrului **s** valoarea **7** deoarece $\mathbf{4}^{-1}$**=1/4=0.25000**… şi suma primelor **3** zecimale este **2+5+0=7**.

27. Considerăm definite următoarele două subprograme:
- **pozmax** – cu doi parametrii: **v** (un tablou unidimensional format din **n** (**1≤n≤100**) numere întregi de cel mult **3** cifre fiecare) şi **p** (un număr natural cuprins între **1** şi **n**); subprogramul returnează poziţia valorii maxime din secvenţa **v[1],...,v[p]**.
- **schimb** – cu doi parametri: **a** şi **b** (numere întregi formate din cel mult trei cifre fiecare); subprogramul interschimbă valorile a două variabile transmise prin intermediul parametrilor **a** şi **b**.

a) Scrieţi numai antetul fiecăruia dintre cele două subprograme.
b) Scrieţi un program care citeşte de la tastatură un număr natural **n** (**1≤n≤100**) şi apoi cele **n** elemente (numere întregi formate din cel mult **3** cifre fiecare) ale unui tablou unidimensional, sortează crescător tabloul apelând subprogramele descrise mai sus şi apoi îl afişează pe ecran, pe o singură linie, elementele tabloului fiind despărţite prin câte un spaţiu.

28. **a)** Scrieţi **numai** antetul subprogramului **divizor**, care primeşte prin intermediul parametrului **n** un număr natural (**n>1**) cu maximum **9** cifre şi returnează prin intermediul parametrului **d** valoarea celui mai mic divizor prim al lui **n**, iar prin intermendiul parametrului **p** puterea la care acest divizor apare în descompunerea în factori primi a numărului **n**.

b) Scrieţi programul Pascal/C care citeşte de la tastatură două numere naturale **n,x (x,n>1)** cu maximum **9** cifre şi verifică dacă **n** este divizibil cu 2^x , folosind apeluri ale funcţiei **divizor** definită la punctul a). Programul afişează **DA** în caz afirmativ şi **NU** în caz contrar.

29. Subprogramul **maxim** primeşte prin intermediul parametrului **A** un vector cu cel mult **100** de numere reale şi prin intermediul parametrului **n** numărul de elemente din vector. El returnează valoarea celui mai mare element din vector, şterge toate apariţiile acestui element din vector şi modifică corespunzător valoarea lui **n**.

a) Scrieţi numai antetul subprogramului **maxim**.
b) Scrieţi programul Pascal/C care citeşte de la tastatură un vector **V** de maximum **100** de numere reale şi afişează, în ordine strict descrescătoare, separate prin spaţiu, numerele care apar în vector. Se vor folosi apeluri ale subprogramului **maxim** definit la punctul a).
De exemplu, pentru vectorul **2,3,1,1,6,2,7,6,1** se afişează **7 6 3 2 1**.

30. Se consideră o funcţie **f** care are ca parametrii un vector **a** cu maximum **50** de elemente întregi, numărul de elemente al vectorului **n**, număr natural (**n≤50**) şi un număr întreg **x**. Funcţia va returna numărul de elemente din vectorul **a** care sunt strict mai mici decât **x**.

a) Scrieţi definiţia completă a funcţiei **f**.

b) Scrieţi un program care citeşte un vector **a** cu **n** elemente întregi (**n≤50**) şi utilizând apeluri utile ale funcţiei **f** numără şi afişează pe ecran câte elemente diferite de **0** sunt în vectorul **a**.

31. **a)** Scrieţi definiţia completă a unui subprogram **inter** care interschimbă conţinutul a două variabile reale transmise prin parametrii **a** şi **b**.

b) Scrieţi declarările necesare şi definiţia completă a unui subprogram **circ** care, folosind doar interschimbări de elemente, permută circular cu o poziţie spre stânga componentele unui vector cu **n** componente reale (**0<n<100**). Parametrii subprogramului sunt: vectorul **v** şi numărul efectiv de elemente ale acestuia, **n**.

De exemplu, pentru **n=5** şi **v=(8,1,10,6,8)**, în urma apelului **circ(v,n)**, conţinutul lui **v** devine **v=(1,10,6,8,8).**
Interschimbările se vor realiza utilizând doar apeluri ale subprogramului **inter**.

32. Subprogramul **majuscula** primeşte prin parametrul **c** un caracter pe care îl transformă în literă mare dacă este literă mică, în caz contrar lăsându-l nemodificat. Subprogramul returnează valoarea **1** dacă parametrul **c** este o vocală (a,e,i,o,u,A,E,I,O,U) şi **0** altfel.

a) Scrieţi doar antetul subprogramului **majuscula**.

b) Fişierul **bac.in** conţine un rând de text format din maximum **1000** de caractere (litere mari sau mici). Scrieţi programul **Pascal/C/C++** care citeşte textul din fişierul **bac.in** şi îl scrie în fişierul **bac.out** modificat, prin apeluri ale subprogramului **majuscula**, astfel: toate caracterele din text să fie litere mari şi după fiecare vocală să apară caracterul semnul exclamării (**!**).
Exemplu: Fişierul **„bac.in"** conţine: **EasdujHIrtoua**
Fişierul **„bac.out"** va conţine**:E!A!SDU!JHI!RTO!U!A!**

33. a) Scrieţi o funcţie care primeşte prin intermediul parametrului **v** un vector cu cel mult **50** elemente numere reale şi prin parametrul **n** o valoare naturală (**2≤n≤50**). Funcţia returnează valoarea **1** dacă primele **n** elementele ale vectorului sunt în ordine strict crescătoare, **2** dacă primele **n** elementele ale vectorului sunt în ordine strict descrescătoare şi **0** dacă primele **n** elemente ale vectorului nu sunt nici în ordine strict crescătoare, nici în ordine strict descrescătoare.

b) Scrieţi un program Pascal/C/C++ care citeşte de la tastatură un număr natural **p** (**1<p<50**) şi apoi cele **p** elemente reale ale unui vector şi care afişează pe ecran numărul de elemente ale celei mai lungi secvenţe din vector care începe cu primul element şi are toaţe elementele ordonate (strict crescător sau strict descrescător). O secvenţa din vector este formată din elemente aflate pe poziţii consecutive. Pentru rezolvarea cerinţei se vor folosi apeluri ale funcţiei de la punctul a).

De exemplu, pentru **p=10** şi şirul: **56 47 34 32 12 4 1 6 11 4** se va afişa **7** deoarece primele **7** elemente din vector sunt în ordine strict descrecătoare.

34. Scrieţi definiţia completă a unui subprogram care creează în memorie un vector **v** cu elemente numere întregi, obţinute prin parcurgerea în ordine a coloanelor unei matrice **a** începând de la prima până la ultima, pe fiecare coloană elementele fiind parcurse de jos în sus. Subprogramul va avea ca parametri: numerele întregi **n** şi **m** ce reprezintă numărul de linii, respectiv de coloane ale matricei date, matricea **a** şi vectorul **v** ce se va construi.
Exemplu : pentru **n=3, m=5** şi matricea :

```
 1 4 -5  0 -7
 4 7 -7 -6  3
-9 2  1 -1 -2
```

atunci **v :(-9,4,1,2,7,4,1,-7,-5,-1,-6,0,-2,3,-7)**

35. Subprogramul **trans** are un singur parametru, notat **c**, de tip **char**; subprogramul va returna tot prin parametrul **c** o valoare obţinută astfel:

- dacă **c** este literă mică a alfabetului englez, litera mare corespunzătoare;
- dacă **c** este literă mare a alfabetului englez, litera mică corespunzătoare;
- dacă **c** nu este literă a alfabetului englez, valoarea lui **c** nemodificată.

a) Scrieţi doar antetul subprogramului **trans**;

b) Scrieţi un program care citeşte de la tastatură un şir de maxim 100 de caractere şi transformă şirul citit înlocuind toate literele mici ale alfabetului englez cu literele mari corespunzătoare şi toate literele mari ale alfabetului englez cu literele mici corespunzătoare, utilizând apeluri ale subprogramului **trans**, apoi afişează şirul rezultat.

Exemplu:

şirul citit este: **Ana-Maria are 3 frati.**
şirul afişat va fi **aNA-mARIA ARE 3 FRATI.**

36. a) Scrieţi un subprogram **f** cu doi parametrii: un şir de caractere **s** cu maxim **50** de caractere şi un număr natural **k**, **1≤k≤50**. Subprogramul va afişa pe ecran primele **k** caractere din şirul **s.** Dacă lungimea şirului **s** este mai mică decât **k**, subprogramul va afişa mesajul **nu este posibil**.

De exemplu: **f('abracadabra', 5)** va afişa **abrac**.

f('abracadabra', 15) va afişa **nu este posibil**.

b) Scrieţi un program care citeşte de la tastatură **n** şiruri de maxim **50** de caractere (**n≤50**) şi le afişează pe ecran utilizând funcţia **f** astfel: pe primul rând, primul caracter din primul şir, pe al doilea rând, primele două caractere din al doilea şir, ..., pe al **n** –lea rând, primele **n** caractere din al **n** -lea şir; acolo unde şirul e mai scurt decât indicele liniei pe care se află, se va afişa mesajul **nu este posibil.**

Exemplu: Dacă **n=5** şi se citesc şirurile de caractere **ionel, aurel, turcu, pop, trasca** se va afişa:

```
i
au
tur
nu este posibil
trasc
```

37. Se consideră subprogramul **elimin** cu doi parametri **s** şi **t**, şiruri de caractere cu maximum **100** de caractere, care primind prin intermediul parametrului **s** un şir de caractere format numai din literele mici ale alfabetului englez, întoarce prin intermediul parametrului **t** şirul obţinut din **s** prin eliminarea tuturor vocalelor (**a, e, i, o, u**).

a) Scrieţi definiţia completă a subprogramului **elimin**.

b) Fişierul **bac.txt** conţine pe fiecare linie câte un şir de maximum **100** de caractere, litere mici ale alfabetului englez. Scrieţi un program care afişează pe câte o linie a ecranului fiecare şir citit din fişier precum şi cel obţinut din acesta prin eliminarea vocalelor, separate printr-un spaţiu. Programul va folosi apeluri utile ale subprogramului **elimin** definit anterior.

De exemplu dacă fişierul **bac.txt** conţine:	Se va afişa:
iepure **oaie** **bcd**	**iepure pr** **oaie** **bcd bcd**

38. a) Scrieţi definiţia unui subprogram **sub** care primeşte prin intermediul parametrului **x** un şir de caractere cu cel mult **200** caractere şi prin intermediul parametrului **c** un caracter; subprogramul determină modificarea şirului **x**, eliminând toate apariţiile caracterului **c** şi returnează numărul ştergerilor efectuate.
De exemplu: pentru şirul de caractere **x="alexandrina"** şi caracterul **c='a'** subprogramul transformă şirul astfel **x="lexndrin"** şi returnează valoarea **3** (s-au efectuat trei eliminări)

b) Scrieţi un program Pascal/C/C++ care citeşte de la tastatură două şiruri de caractere cu cel mult **200** de caractere fiecare şi care afişează pe ecran toate caracterele comune celor două şiruri, folosind apeluri ale subrogramului **sub**. Fiecare caracter se va afişa o singură dată, caracterele afişându-se separate printr-un spaţiu.
De exemplu pentru şirurile: **"matrice"** şi **"principal"** se va afişa: **r i c a** nu în mod obligatoriu în această ordine.

39. Subprogramul **ordalfa** primeşte prin parametrul formal **a** un şir cu maximum **255** de caractere şi returnează valoarea **1**, dacă şirul **a** este ordonat alfabetic (are caracterele în ordine crescătoare a codurilor **ASCII**), sau **0** în caz contrar.

a) Scrieţi definiţia completă a subprogramului **ordalfa**.

b) Fişierul **bac.txt** conţine pe primul rând o valoare naturală **n** (**n≤10**), iar pe următoarele **n** rânduri, **n** şiruri de caractere (câte unul pe fiecare rând), cu maximum **127** caractere fiecare. Scrieţi programul Pascal/C/C++ care determină numărul de perechi de şiruri de caractere **(x,y)**, aflate pe rânduri consecutive în fişierul **bac.txt**, şi care au proprietatea că şirul obţinut din concatenarea (lipirea) lui **y** la **x**, sau a lui **x** la **y** este ordonat lexicografic.
Exemplu: pentru fişierul **bac.txt**, din caseta alăturată, se va tipări **2**, deoarece următoarele perechi respectă proprietatea: **(adgl,rs)** şi **(mop,rs)**.

```
4
adgl
rs
mop
vnu
```

40. Se consideră subprogramul **max_diviz** care are doi parametri, **a** şi **b** (numere naturale având maximum **9** cifre fiecare) şi care returnează **cel mai mare divizor dintre divizorii comuni de câte două cifre** ai numerelor **a** şi **b**. Dacă cele două numere nu au un astfel de divizor, subprogramul va returna **1**.

a) Scrieţi definiţia completă a subprogramului **max_diviz**.

b) Scrieţi programul Pascal/C/C++ care citeşte de la tastatură un număr natural nenul **a** cu maximum **9** cifre, şi determină, folosind apeluri ale subprogramului **max_diviz,** cel mai mare număr de două cifre care divide pe **a**. Se va afişa pe ecran numărul astfel determinat sau mesajul: **NU exista divizor de 2 cifre**, în cazul în care **a** nu are nici un divizor de două cifre. De exemplu: pentru **a=160** se va afişa **80**.

41. Subprogramul **cifre_pare** primeşte ca parametru un număr natural **n** având cel mult **9** cifre şi verifică dacă numărul **n** are toate cifrele pare.

a) Scrieţi definiţia completă a subprogramului **cifre_pare**.

b) Scrieţi programul care folosind apelul subprogramului **cifre_pare**, afişează pe ecran numerele naturale mai mici decât **10000**, care au toate cifrele pare.

42. Se citesc de la tastatură trei numere naturale nenule, **a**, **b** şi **m**, de cel mult 3 cifre fiecare. Scrieţi programul care afişează pe ecran, primii **m** multipli comuni ai celor două numere **a** şi **b**.

Exemplu:pentru **a=180,b=120** şi **m=5**

se vor afişa numerele: **360,720,1080,1440,1800**

43. Şirul de numere **1,2,3,1,3,1,0,4, ...** este definit prin relaţia alăturată, în care s-a notat cu **x%y** restul împărţirii numărului natural **x** la numărul natural **y**.

$$f_n = \begin{cases} n, & pentru\ n \le 3 \\ (f_{n-1} * f_{n-2} + f_{n-3} + f_{n-2})\%n, & pentru\ \ n \ge 4 \end{cases}$$

a) Descrieţi în limbaj natural o metodă eficientă (ca timp de executare şi spaţiu de memorare) prin care se poate determina elementul din şir aflat pe o poziţie dată, **n**. Explicaţi în ce constă eficienţa metodei alese (**4-8** rânduri).

b) Scrieţi definiţia completă a unui subprogram **F** care are un singur parametru, **n**, prin intermediul căruia primeşte o valoare naturală cu cel mult **3** cifre, astfel încât prin instrucţiunea:

```
write(F(n));  | cout<<F(n); /printf("%d",F(n));
```

să se afişeze valoarea celui de-al **n**-lea termen din şir, utilizând algoritmul descris la punctul a).

Exemplu: Instrucţiunea

`write(F(7));`	`cout<<F(7);` `/printf("%d",F(7));`

va afişa valoarea `0`.

44. Un număr natural nenul se numeşte **echilibrat** dacă numărul cifrelor pare din scrierea lui este egal cu numărul cifrelor impare. De exemplu numerele **1234** şi **1212** sunt echilibrate, iar numărul **123** nu este echilibrat.

Realizaţi următoarele cerinţe utilizând limbajul **Pascal/C/C++**:

a) Scrieţi definiţia completă a unui subprogram **sub** care primeşte prin intermediul parametrului **n** un număr natural cu cel mult **8** cifre şi care returnează valoarea **1** dacă numărul **n** este echilibrat şi returnează **0** în caz contrar. De exemplu **sub(1234)** returnează **1**, iar **sub(123)** returnează **0**.

b) Scrieţi un program care citeşte din fişierul text **bac.txt** un număr natural **n** (**n<1000**) şi un şir de **n** numere naturale mai mici decât 30000; programul determină şi afişează pe ecran secvenţa de lungime maximă de termeni ai şirului, aflaţi pe poziţii consecutive şi care sunt numere echilibrate, separaţi prin câte un caracter spaţiu, folosind apeluri ale subprogramului de la punctul **a)**.

Exemplu: Dacă fişierul **bac.txt** conţine numerele: **8, 124, 1724, 34, 6, 1681, 1241, 1221, 445** se afişează **1681 1241 1221.**

45. Se consideră subprogramul **numar**, care:
- primeşte prin intermediul unicului său parametru, **a**, un număr natural de cel mult **4** cifre;
- returnează numărul divizorilor lui **a**;

a) Scrieţi numai antetul funcţiei **numar.**

b) Scrieţi declarările de date şi programul principal în care, folosind numai apeluri ale subprogramului **numar,** se verifică dacă un număr natural **k** (**1<k<10000**) citit de la tastatură este prim. Programul va afişa pe ecran mesajul **DA,** dacă numărul **k** este prim, sau mesajul **NU,** în caz contrar.

46. Se consideră subprogramul **fibo**, care:
- primeşte prin intermediul parametrului **a** un număr natural nenul de cel mult **4** cifre;
- returnează valoarea celui mai mare termen al şirului lui Fibonacci mai mic sau egal cu **a** (un termen general al şirului lui Fibonacci, f_n, este definit după regulile: $f_1=f_2=1$ şi $f_n=f_{n-1}+f_{n-2}$ pentru orice $n>2$)

a) Scrieţi numai antetul subprogramului **fibo**.

b) Scrieţi declarările de date şi programul principal în care se afişează o decompunere a unui număr natural nenul de cel mult **4** cifre **n** citit de la tastatură ca sumă de termeni distincţi ai şirului lui Fibonacci, folosind apeluri ale subprogramului **fibo**. De exemplu, dacă se citeşte **n=17**, un rezultat corect afişat de program este: **1 3 13**.

47. Se consideră funcţia **divizor**, care:
- primeşte prin intermediul parametrului **a** un număr natural de cel mult **9** cifre, **a>1**;
- returnează valoarea celui mai mare divizor al lui **a** diferit de **a** (de exemplu, dacă **a=27**, funcţia va returna **9**)

a) Alegeţi o metodă eficientă de determinare a divizorului cerut, descriind în limbaj natural metoda folosită şi explicând în ce constă eficienţa acesteia (cel mult **6** rânduri).

b) Scrieţi definiţia completă a funcţiei **divizor**.

c) Scrieţi declarările de date şi programul principal în care se verifică dacă un număr natural de cel mult **9** cifre **n (n>1)** citit de la tastatură este prim, folosind apeluri ale subprogramului **divizor**.

48. **a)** Scrieţi **numai** antetul funcţiei **maxim**, care primeşte prin intermediul parametrului **A** o matrice pătrată cu **10** linii şi **10** coloane cu elemente reale şi prin intermediul parametrilor **i1** şi **i2** două numere naturale ce reprezintă indicii a două linii din matricea **A** şi returnează diferenţa maximă dintre două elemente, aflate unul pe linia **i1** şi celălalt pe linia **i2**.

b) Scrieţi programul Pascal/C care citeşte de la tastatură o matrice pătrată cu **10** linii şi **10** coloane de numere reale denumită **A** şi apoi calculează şi afişează diferenţa maximă dintre două elemente ale matricei **A** aflate pe linii **diferite**, folosind apeluri ale funcţiei **maxim** definită la punctul a).

De exemplu, pentru matricea **A**:

```
0 2 4 ..18
1 1 1 ...1
2 2 2 ...2
...
9 9 9 ...9
```

se va afişa **17**.

49. Subprogramul **f** primeşte prin intermediul parametrului **n** un număr natural cuprins între **1** şi **10.000** şi returnează cel mai mare termen din şirul lui Fibonacci strict mai mic decât **n**.

a) Scrieţi definiţia completă a subprogramului **f**.

b) Scrieţi programul care citeşte un tablou unidimensional de numere naturale nenule mai mici decât **10.000**, apoi înlocuieşte fiecare element al tabloului cu cel mai mare termen din şirul lui Fibonacci strict mai mic decât el şi afişează elementele tabloului astfel obţinut pe o singură linie a ecranului, separate prin spaţiu.Se vor folosi apeluri utile ale subprogramului **f** definit la punctul **a)**. De exemplu, dacă se citeşte tabloul **(20,2,87,3,120)** atunci se va afişa: **13 1 55 2 89.**

50. a) Două subprograme **s1** şi **s2** primesc ca parametri câte un şir, **x**, de cel mult **100** numere întregi strict mai mici decât **1000**, şi câte două valori **p** şi **q** (**0≤p≤q<101**). Subprogramul **s1** returnează maximul elementelor şirului **x**

situate între poziţiile **p** şi **q**, inclusiv **p** şi inclusiv **q**, iar subprogramul **s2** returnează prin parametrul **min** minimul elementelor şirului **x** situate între poziţiile **p** şi **q**, inclusiv **p** şi inclusiv **q**. Scrieţi definiţiile complete ale subprogramelor anterioare.

b) Scrieţi programul care citeşte de la tastatură două tablouri unidimensionale **a** şi **b** de cel mult **100** numere întregi strict mai mici decât **1000** şi afişează pe ecran mesajul „**DA**" în cazul în care fiecare element din **a** este strict mai mic decât toate elementele din **b**, respectiv „**NU**", în caz contrar. Se vor folosi apeluri utile ale subprogramelor precizate anterior.

51. Scrieţi un program **Pascal/C/C++** care citeşte de la tastatură un număr natural **n(2<=n<=100)**, apoi **n** numere naturale de cel mult **9** cifre fiecare şi afişează numerele ordonate descrescător după suma cifrelor. Dacă două sau mai multe numere au aceeaşi sumă a cifrelor, vor fi afişate în ordinea în care au fost citite. Pentru **n = 6** şi numerele **124 800 7000 578004 789 312** se va afişa **578004 789 800 124 7000 312.**

52. Se citesc 2 numere naturale nenule **m,n (2<m,n<20).** Să se scrie programul **PASCAL/C/C++** care construieşte în memorie o matrice **A** cu **m** linii şi **n** coloane cu proprietatea că elementul **A[i][j]** este egal cel mai mare divizor comun dintre numerele **i** şi **j** . Matricea se va afişa pe ecran, câte o linie a matricei pe câte o linie a ecranului, elementele fiecărei linii fiind separate prin spaţii.

De exemplu pentru **m=3** şi **n=4** se va afişa matricea următoare:

```
1  1  1  1
1  2  1  2
1  1  3  1
```

53. Se citeşte de la tastatură o valoare naturală **m** (**2<=m<=100**). Scrieţi programul **Pascal/C/C++** care construieşte în memorie şi apoi afişează pe ecran matricea **a** cu **m** linii şi **m** coloane de numere întregi construită după următoarea regulă: elementul de pe linia **i** şi coloana **j** a matricii (**1<=i,j<=m**) este:

- **1** dacă **i*i+j*j** este număr prim;
- **2** dacă **i*i+j*j** nu este număr prim, dar suma **i+j** este număr par mai mare decât **2**;
- **0** în rest.

De exemplu pentru **m = 4** se va afişa:

```
1 1 2 1
1 2 1 2
2 1 2 0
1 2 0 2
```

54. a) Scrieţi definiţia completă a unui subprogram **elimin** care:
- primeşte ca parametri două şiruri **s1** şi **s2** de maximum **250** de litere mici ale alfabetului englez;

- returnează prin parametrul **s** şirul de litere distincte care apar fie în şirul **s1**, fie în şirul **s2**, fie în ambele şiruri.

De exemplu, dacă iniţial în **s1** este memorat şirul **aebacdbaced**, iar în **s2** este memorat **aeeffgg** în final în **s** va fi memorat **aebcdfg**, nu neapărat în această ordine.

b) Scrieţi declarările de date şi programul principal în care se citeşte de la tastatură un număr **n**(**1<n<20**) şi apoi **n** şiruri de caractere formate din maxim **250** de litere mici ale alfabetului şi afişează pe ecran un şir de caractere distincte format din toate literele care apar cel puţin în unul din şirurile citite, utilizând apeluri ale subprogramului **elimin.**

De exemplu, dacă **n=4** şi cele **4** şiruri sunt **xabc**, **yyaad**, **abcd**, **ccddzz**, se va afişa pe ecran **abcdxzy**, nu neapărat în această ordine.

55. Se consideră subprogramul **aparitii** care primeşte prin intermediul parametrului **s** un şir de maximum **100** de caractere iar prin intermediul parametrului **x** un caracter şi returnează numărul de apariţii ale caracterului **x** în şirul **s**.

a) Scrieţi definiţia completă a subprogramului **aparitii**.

b) Scrieţi programul Pascal/C care citeşte de la tastatură două şiruri de maximum **100** de litere mici şi verifică, utilizând apeluri ale funcţiei **aparitii** dacă cele două şiruri sunt anagrame (conţin aceleaşi litere, ordinea acestora fiind **diferită**). Se cere afişarea mesajului **anagrame** în caz afirmativ şi a mesajului **nu sunt anagrame** în caz contrar.

De exemplu, pentru şirurile **lada** şi **dala** se afişează **anagrame**.

4.2. Subprograme recursive

4.2.1. Teste grilă (Pascal | C / C++)

1. Ce valoare va fi returnată la apelul **f(20)**?

```
function
f(n:integer):integer;
Begin if n<=1 then f:=n
       else f:=f(n-2)+n
end;
```

```
int f(int n)
{
  if (n<=1) return n;
    else return f(n-2)+n;
}
```

a. 210 **b.** 110 **c.** 0 **d.** 20

2. Pentru definiţia următoare a subprogramului **f**, stabiliţi ce valoare returnează apelul **f(8,10)**?

```
Function f(x,y:byte):longint;
Begin
 if (x=y) or (x=0) then
  f := 1
 else
  f:= f(x,y-1) + f(x-1,y-1)
End;
```

```
long f(int x, int y)
{if (x==y || x==0)
    return 1;
 else
   return
     f(x,y-1) + f(x-1,y-1);
}
```

a. 50 **b.** 45 **c.** 40 **d.** 55

3. Pentru definiţia următoare a subprogramului **f**, stabiliţi ce valoare returnează apelul **f(6,3)**?

```
Function f(n,k:integer):longint;
Var s,i : longint;
Begin
 if (n=k) or (k=1) Then f := 1
 else if n<k Then f := 0
      else Begin s := 0;
           For i:=1 to k do
             s = s + f(n-k,i);
           f := s
      End
End;
```

```
long f(int n, int k)
{ long s=0,i;
 if (n==k || k==1)
    return 1;
 if (n<k) return 0;
 for(i=1;i<=k;i++)
   s = s + f(n-k,i);
 return s;
}
```

a. 3 **b.** 1 **c.** 2 **d.** 4

4. Pentru definiţia următoare a subprogramului **f**, stabiliţi ce valoare returnează apelul **f(7)**?

```
function  f(int n) : longint;
begin
 if n=0 then f := 1
 else
   if n=1 then f := 4
     else f := f(n-1) - f(n-2)
end;
```

```
long f(int n)
{
 if (n==0) return 1;
 else
   if (n==1) return 4;
    else
  return f(n-1) - f(n-2);
}
```

a. 1 **b.** 3 **c.** -3 **d.** 4

5. Ce se afişează ca urmare a apelului **f(2)**; , dacă subprogramul **f** are declaraţia următoare?

```
procedure f(n:integer);
var j:integer;
begin
  if n>0 then
   for j:=1 to n do
    begin  write(j);
           f(n-1)
    end
end;
```

```
void f(int n)
{
  int j;
  if (n>0)
   for (j=1;j<=n;j++)
    {
     cout<<j;/printf("%d",j);
     f(n-1);
    }
}
```

a. 1122 **b.** 112 **c.** 121 **d.** 1121

6. Fie subprogramul **f** definit mai jos şi **a** o variabilă de tip întreg. Dacă în urma apelului **f(f(a))** valoarea returnată de funcţie a fost **55**, atunci valoarea variabilei **a** a fost:

```
function f(n:integer):integer;
begin
  if (n=0) then f:=0
  else f:=n+f(n-1)
end;
```

```
int f(int n)
{
  if (n==0) return 0;
  else return n+f(n-1);
}
```

a. 10 **b.** 4 **c.** 3 **d.** 5

7. Fie subprogramul **f** definit mai jos şi **a** o variabilă de tip întreg. Dacă în urma apelului **f(a)** valoarea returnată de funcţie a fost **153**, atunci valoarea variabilei **a** a fost:

```
function f(n:integer):integer;
begin
  if (n=0) then f:=0
  else f:=n+f(n-1)
end;
```

```
int f(int n)
{
  if (n==0) return 0;
  else return n+f(n-1);
}
```

a. 18 **b.** 31 **c.** 20 **d.** 17

8. Fie subprogramul **f** definit mai jos. Care este valoarea expresiei **f(4,12)**?

```
function f(x,y:integer):integer;
begin
  if (x>y) then f:=0
  else
   if (x mod 2<>0) then
        f:=1+f(x+1,y)
   else f:=f(x+1,y)
end;
```

```
int f(int x,int y)
{ if (x>y) return 0;
  else
   if (x%2!=0)
      return 1+f(x+1,y);
   else return f(x+1,y);
}
```

a. 3 **b.** 2 **c.** 4 **d.** 5

9. Considerând subprogramul **f**, definit mai jos, stabiliţi ce valoare are expresia **f(10,30)**?

```
function f(x,y:integer):integer;
begin
 if (x>y) then f:=0
 else if (x mod 2=0)
       then f:=1+f(x+1,y)
       else f:= f(x+1,y)
end;
```

```
int f(int x,int y)
{ if (x>y) return 0;
  else if (x%2==0) return
1+f(x+1,y);
       else return
f(x+1,y);
}
```

a. 20 **b.** 11 **c.** 10 **d.** 15

10. Pentru definiţia următoare a subprogramului **f**, stabiliţi ce valoare returnează apelul **f(30,4)**?

```
function f(x,y:integer):integer;
begin
    if(x<y) then f:=0
    else f:=y+f(x-y,y);
end;
```

```
int f(int x,int y)
{
    if (x<y) return 0;
    else return  y+f(x-y,y);
}
```

a. 28 **b.** 30 **c.** 32 **d.** 7

11. Pentru definiţia următoare a subprogramului **f**, ce se va afişa în urma apelului **f(14663)**?

```
function f(n:integer):integer;
begin
 if n=0 then f:=0
 else if n mod 2<>0
      then f:=1+f(n div 10)
         else f:=f(n div 10)
end
```

```
int f(int n)
{
 if(n==0)return 0;
 if(n%2)return 1+f(n/10);
 return f(n/10);
}
```

a. 5 **b.** 0 **c.** 2 **d.** 3

12. Pentru definiţia următoare a subprogramului **f**, ce valoare se va returna la apelul **f(14625)**?

```
function f(n:integer):integer;
var m:integer;
begin
if n=0 then f:=-1
```

```
int f(int n)
{
  if(n==0)return -1;
  int m=f(n/10);
```

```
else
begin m:=f(n div 10);
if(n mod 2<>0)or(m>n mod 10)
    then f:=m
    else f:=n mod 10
  end
end;
```

```
  if(n%2||m>n%10)return m;
  return n%10;
}
```

a. -1 **b.** 2 **c.** 6 **d.** 1

13. Pentru definiţia următoare a subprogramului **f**, ce se va returna la apelul **f(20400)**?

```
function f(n:integer):integer;
begin
 if n<>0 then begin
   if n mod 10<>0
     then f:=f(n div 10)
     else f:=1+f(n div 10)
 end
 else f:=0
end;
```

```
int f(int n)
{ if(n!=0)
  {
  if(n%10!=0)return f(n/10);
  return 1+f(n/10);
  }
 return 0;
}
```

a. 6 **b.** 5 **c.** 0 **d.** 3

14. Pentru definiţia următoare a subprogramului **f**, ce se va afişa la apelul **f(27524)**?

```
procedure f(n:integer);
begin
 if n<>0 then
   if n mod 2=1 then
    begin
     write(n mod 10);
     f(n div 10)
    end
   else  begin
     f(n div 10); write(n mod 10)
         end
end;
```

```
void f(int n)
{ if(n)
 { if(n%2)
    {printf("%d",n%10);
          /cout<<n%10;
     f(n/10);}
   else
     {f(n/10);
      printf("%d",n%10);
      /cout<<n%10;
      }
 }}
```

a. 27524 **b.** 75422 **c.** 57224 **d.** 42572

15. Pentru funcţia recursivă următoare stabiliţi care este valoarea returnată în urma apelului **f(4)**?

```
function f (x:integer):integer;
begin if x<=1 then f:=x+1
else f:=f(x-1)+2*f(x-2) end;
```

```
int f ( int x)
{ if(x<=1) return x+1;
else
return f(x-1)+2*f(x-2);}
```

a. 16 **b.** 15 **c.** 4 **d.** 11

16. Câte caractere ***** se afişează în urma apelului **stea(3)?**

```
procedure stea (x:integer);
var i:integer;
begin if x>0 then
 begin stea(x-1);
 for i:=1 to x do write('*')
  end
end;
```

```
void stea (int x)
{int i;
  if(x>0)
 {stea(x-1);
 for(i=1;i<=x;i++)
  printf("*");|cout<<"*";
 }}
```

a. 6 **b.** 3 **c.** 9 **d.** 12

17. Pentru definiţia subprogramului următor stabiliţi ce se afişează la apelul `f(1,5)`.

```
procedure f(i,j:integer);
begin
if i<=j then
 if i+j<5 then
begin write(i);
      f(i+1,j)end
else
begin write(j);f(i,j-1) end
end;
```

```
void f(int i, int j)
{if(i<=j)
   if(i+j<5){
    printf("%d",i); | cout<<i;
    f(i+1,j);}
   else {
    printf("%d",j); | cout<<j;
    f(i,j-1);}}
```

a. 54132 **b.** 12345 **c.** 51423 **d.** 54321

18. Pentru definiţia subprogramului următor stabiliţi ce se afişează la apelul `f(5,1)`.

```
procedure f(i,k: integer);
begin
if k<=4 then begin
               write(i*k);
               f(i-1,k+1)
             end
end;
```

```
void f(int i, int k)
{if(k<=4){
printf("%d",i*k);|cout<<i*k;
       f(i-1,k+1);
         }
}
```

a. 1248 **b.** 5898 **c.** 1234 **d.** 5488

19. Pentru definiţia subprogramului următor stabiliţi ce se afişează la apelul `f(5,1)`.

```
procedure f(n,k:integer);
begin
if k<=n then
  begin write(n-k);f(n,k+1)
  end
end;
```

```
void f(int n, int k)
{if(k<=n){
           printf("%d",n-k);
               |cout<<n-k;
           f(n,k+1);
        }}
```

a. 12345 **b.** 01234 **c.** 43210 **d.** 54321

20. Pentru definiţia următoare a subprogramului `f`, ce se valoare are expresia `f(245284003)`?

```
function f(n:longint):integer;
begin
 if n=n mod 10 then f:=n
 else f:=f(n div 10)
end;
```

```
int f(long n){
 if(n==n%10) return n ;
 return f(n/10);
}
```

a. 3 **b.** 0 **c.** 2 **d.** 9

21. Ce valoare trebuie să primească la apel parametrul formal **n**, pentru ca funcţia următoare să returneze valoarea **21**?

```
function ex(n:integer):integer;
begin
if n=0 then ex:=0
else ex:=n+ex(n-1) end;
```

```
int ex(int n)
{
if(n==0) return 0;
else return n+ex(n-1);}
```

a. 7 **b.** 8 **c.** 4 **d.** 6

22. Fie subprogramul definit mai jos. Ce se afişează în urma apelului **P(3)**?

```
procedure P(x:integer);
begin if x<>0 then P(x-1)
      write(x)
end;
```

```
void P(int x){
if(x)P(x-1);
cout<<x;  |  printf(˝%d˝,x);
}
```

a. 3 **b.** 123 **c.** 3210 **d.** 0123

23. În subprogramul recursiv următor se consideră că vectorul de numere întregi **v** este declarat global. Pentru care dintre vectorii **v** următori, la apelul **star(0)**, se afişează **7** asteriscuri?

```
procedure star(i:byte);
begin
  if i<=10 then begin
    write('*');
    if v[i]=i then star(i+2)
    else star(i+1)
  end
end;
```

```
void star(int i)
{
 if(i<10){
  printf("*"); | cout<<'*';
  if(v[i]==i+1)star(i+2);
  else star(i+1);
 }
}
```

a. (1,4,3,2,1,6,5,4,3,10) **b.** (1,2,3,4,5,6,7,8,9,10)
c. (3,2,1,4,3,6,7,2,9,2) **d.** (10,9,8,7,6,5,4,3,2,1)

24. Pentru definiţia următoare a subprogramului **ex**, stabiliţi ce se afişează la apelul **ex(120)**?

```
procedure ex( x:byte);
begin
  if x<>0 then begin
     write(x mod 10);
     ex(x div 10)
  end
end;
```

```
void ex(int x)
{if(x!=0) {
  printf("%d",x%10);
        / cout<<x%10;
  ex(x/10);
 }
}
```

a. 021 **b.** 012 **c.** 120 **d.** 21

25. Pentru definiţia următoare a subprogramului **ex**, stabiliţi ce se afişează la apelul **ex(120)**?

```
procedure ex( x:byte);
begin
   write(x mod 10);
   if x<>0 then
            ex(x div 10)
end;
```

```
void ex(int x)
{
  printf("%d",x%10);
          /cout<<x%10;
  if(x!=0) ex(x/10);
}
```

a. 0120 **b.** 021 **c.** 210 **d.** 0210

26. Pentru definiţia următoare a subprogramului **ex**, stabiliţi ce se afişează la apelul **ex(2,10)**?

```
procedure ex( i,j:byte);
begin
if i<=j then
   begin
     write(i);
     ex(i+1,j-1);
     if i mod 2<>0
        then write(9-j)
   end
end;
```

```
void ex(int i,int j)
{ if(i<=j) {
    printf(”%d”,i);
      / cout<<i;
    ex(i+1,j-1);
    if(i%2!=0)
      printf(”%d”,9-j);
        / cout<<9-j;
  }
}
```

a. 2345620 **b.** 246357 **c.** 2345646 **d.** 234520

27. Se ştie că variabila **a** de tip **integer/int** memorează valoarea **0**. Pentru definiţia următoare a subprogramului **ex**, ce valoare va avea variabila **a** în urma executării apelului **ex(10542,1821,a)**?

```
procedure ex(n,m: word;
             var z:word);
var c:byte;
begin
 if n+m>0 then  begin
          c:=n mod 10;
   if m mod 10>c then
          c:=m mod 10;
   z:=z*10+c;
   ex(n div 10,m div 10,z) end
end;
```

```
void ex(int n, int m, int &z)

{ int c;
  if(n+m>0)
    {  c=n%10;
       if(m%10>c)c=m%10;
       z=z*10+c;
       ex(n/10,m/10,z);
     }
}
```

a. 12500 **b.** 24811 **c.** 11248 **d.** 2481

28. Se consideră următoarea funcţie recursivă apelată numai pentru numere naturale nenule:

```
function f(a, b:integer):integer;
begin
 if a<b then f:=a
        else f:=f(a-b,b);
end;
```

```
int f(int a, int b)
{ if (a<b) return a;
    else return f(a-b,b);
}
```

Care din următoarele funcţii este echivalentă cu funcţia dată?

a.
```
function
f(a,b:integer):integer;
begin  f:= a * b; end;
```

b.
```
functionf(a,
b:integer):integer;
begin  f:= a mod b; end;
```

c.
```
function                f(a,
b:integer):integer;
begin  f:= a-b+1; end;
```

d.
```
function                f(a,
b:integer):integer;
begin  f:= a div b; end;
```

a.
```
int f(int a, int b)
      { return a*b; }
```

b.
```
int f(int a, int b)
      { return a%b; }
```

c.
```
int f(int a, int b)
      { return a-b+1; }
```

d.
```
int f(int a, int b)
      { return a/b; }
```

29. Ce se afişează în urma apelului **f(247)**, dacă subprogramul **f** are următoarea definiţie:

```
procedure f(n:integer);
begin
 if n<>0 then begin
      f(n div 10);
      write(n mod 10, ' ')
     end;
end;
```

```
void f(int n)
 { if (n!=0)
     { f(n/10);
       cout<<n%10<<" ";
      / printf("%d ", n%10);
     }
 }
```

a. 7 4 2 **b.** 4 2 7 **c.** 2 7 4 **d.** 2 4 7

30. Fie subprogramul recursiv:

```
procedure f(n:integer);
begin
 if n<>0 then begin
    write(n mod 10, ' ');
    f(n div 10)
  end;
end;
```

```
void f(int n)
 { if (n)
    { cout<<n%10<<" ";
      / printf("%d ", n%10);
      f(n/10);}
 }
```

Ce afişează procedura la apelul **f(247)**?

a. 7 4 2 **b.** 2 4 7 **c.** 2 7 4 **d.** 4 2 7

31. Se defineşte subprogramul următor; ce se afişează în urma apelului **f(5)**?

```
procedure f( n:integer);
begin
 if n>0 then
   begin
     f(n-1); write(n,' ' );
   end
end;
```

```
void  f (int n)
{ if (n>0)
 { f(n-1);
   cout<<n<<" ";
   /printf("%d ",n);
 }
}
```

a. 5 4 3 2 1 **b.** 5 4 3 2 1 0 **c.** 5 **d.** 1 2 3 4 5

32. Se consideră subprogramul **f** cu definiţia următoare. Ce se afişează în urma apelului **f(6)**?

```
procedure f(x:integer);
begin
 if x>=0
  then begin  write(x);f(x-1);
    if x mod 2=0 then write(x)
       end
end;
```

```
void f(int x)
 {if (x>=0)
   {cout<<x; / printf("%d", x);
    f(x-1);
    if(x%2==0)
     cout<<x; / printf("%d", x);
}}
```

a. 65432100246 **c.** 0123456

b. 65432106420 **d.** 6543210

33. Câte caractere „*” se vor afişa în urma executării subprogramului următor, la apelul **Scrie(1,1)**?

```
procedure Scrie(i,j:integer);
begin
  if i <= 3 then
       if j <= i then
       begin
              write('*');
              Scrie(i, j+1)
       end
       else Scrie(i+1, 1)
end;
```

```
void Scrie(int i, int j)
{
  if(i <= 3)
    if(j <= i)
     { rintf("*");|cout<<'*';
       Scrie(i, j+1);
       }
       else Scrie(i+1, 1);
}
```

a. 5 **b.** 6 **c.** 3 **d.** 10

34. Se consideră subprogramul recursiv următor. Ce se afişează la apelul **tip(4)**?

```
procedure tip(i:byte)
begin
  write('X');
  if i>0 then
   if i mod 2=0
    then tip(i div 2)
    else tip(i-1)
end;
```

```
void tip(int i)
{
 printf("X");      |
       cout<<'X';
 if (i>0)
   if (i%2==0)tip(i/2);
   else tip(i-1);
}
```

a. XXXXXX **b.** XXXX **c.** XX **d.** XXX

35. Ce se va afişa în urma apelului **F(123),** pentru funcţia **F** definită mai jos:

```
function F(n:integer):boolean;
begin
       F := false;
       if (n=0)or F(n div 10)
        then begin
             write(n mod 10);
              F := true
         end
end;
```

```
int F(int n)
{
 if (n==0 || F(n/10) )
       { cout << n%10;
         return 1;
       }
 return 0;
}
```

a. 0123 **b.** 123 **c.** 321 **d.** 3210

36. Ce valoare va avea în urma apelului **X(5639)** variabila globală **i** de tip întreg, dacă înainte de apel **i=0** şi funcţia **X** este definită mai jos?

```
function X(n:longint):integer;
begin
 i := i + 1;
 if (n = 0) then X:= 0
    else
 X:=X(n div 10)+X(n div 100)
end;
```

```
int X(long int n)
{i++;
 if (n == 0) return 0;
 else
   return X(n/10)+X(n/100);
}
```

a. 15 **b.** 7 **c.** 8 **d.** 16

37. Fie subprogramul **f** definit mai jos. Ce se afişează în urma apelului **f(523)?**

```
procedure f(x:integer);
 begin
  write(x mod 10);
  if x>9 then  f(x div 10);
  write(x mod 10)
 end;
```

```
void f(int x)
{
  cout<<x%10; / prinf("%d",x%10);
  if(x>9) f(x/10);
  cout<<x%10; / prinf("%d",x%10);
}
```

a. 325523 **b.** 325325 **c.** 325 **d.** 523523

38. Pentru o valoare naturală mai mare decât **1** memorată în variabila globală **n**, subprogramul recursiv următor afişează cel mai mare divizor al lui **n**, mai mic decât **n**, la apelul **divi(n)**. Cu ce expresii trebuie completate punctele de suspensie?

```
procedure divi(i:longint);
begin
 if ... =0
     then writeln(...)
     else divi(i-1)
end;
```

a. **n mod i** şi **i**
b. **n mod (i-1)** şi **i-1**
c. **n mod (i-1)** şi **i**
d. **n mod i** şi **i-1**

```
void divi(long i)
{ if (... ==0)
    cout<<...;
       / printf("%ld",...);
  else divi(i-1);

}
```

a. **n%i=0** şi **i**
b. **n%(i-1)** şi **i-1**
c. **n%(i-1)=0** şi **i**
d. **n%i** şi **i-1**

39. Fie funcţia recursivă:

```
function
f(i,j:integer):integer
begin
 if (i=1) or (j=1) then f:= 1
     else
     if  i<j then
       f:= f(i,j-1)+f(i-1,j-1)
             else
       f:= f(i-1,j)+f(i-1,j-1)
end;
```

```
int f(int i, int j)
 { if (i==1 || j==1) return 1;
    else
    if (i<j)
    return
      f(i,j-1)+f(i-1,j-1);
    else
    return f(i-1,j)+f(i-1,j-1);
 }
```

Ce returnează funcţia la apelul **f(2,3)**?

a. 5 **b.** 3 **c.** 4 **d.** 2

40. Se consideră funcţia recursivă:

```
function f(n:integer):integer;
begin
 if n=0 then f:=1
    else
     if n mod 2=0
      then  f:=f(n div 10)
      else  f:=0;
end;
```

```
int f(int n)
{
if (n==0) return 1;
    else
       if (n%2==0)
           return f(n/10);
           else return 0;
}
```

Care dintre instrucţiunile următoare afişează 1?

a. `write(f(2212));`
b. `write(f(10));`
c. `write(f(9426));`
d. `write(f(2426));`

a. `cout<<f(2212); / printf("%d", f(2212));`
b. `cout<<f(10); / printf("%d", f(10));`
c. `cout<<f(9426); / printf("%d", f(9426));`
d. `cout<<f(2426); / printf("%d", f(2426));`

41. Pentru subprogramul **rec** cu definiţia următoare, ce se va afişa în urma apelului **rec(35,4)**?

```
procedure rec( x, y: word );
begin
if x>y then rec(x div y, y);
write( x mod y)
end;
```

```
void rec(unsigned x,unsigned y)
{ if(x>y) rec(x/y,y);
  printf("%u",x%y);
    /cout<<(x%y);
}
```

a. 302 **b.** 203 **c.** 100011 **d.** 83

42. Se consideră funcţia următoare. Ce se va afişa pe ecran la apelul **f(13)**?

```
procedure f(x:integer);
begin
 if x>1 then f(x div 2);
 write(x mod 2)
end;
```

```
void f(int x)
{ if (x>1) f(x/2);
 printf("%d",x%2);| cout<<x%2;
}
```

a. 0101 **b.** 1100 **c.** 1101 **d.** 1011

43. Ce se va afişa pe ecran la apelul **p(630,2,1)**?

```
procedure p(x,y,k:integer);
begin
if x>1 then
 if x mod y =0 then begin
  if k=1 then write(y,' ');
  p(x div y,y,k+1)
 end
 else p(x,y+1,1)
end;
```

```
void p(int x,int y,int k)
{if(x>1)
   if(x%y==0)
     {if(k==1)
       printf("%d ",y);
          | cout<<y<<" ";
        p(x/y,y,k+1);
     }
   else p(x,y+1,1);
}
```

a. 2 **b.** 2 3 3 5 7 **c.** 2 3 5 7 **d.** 2 1 3 2 5 1 7 1

44. Pentru definiţia următoare a subprogramului **p**, stabiliţi ce returnează funcţia la apelul **p(9876543)?**

```
function f(n:longint):integer;
begin
 if n<>0 then
   if n mod 2=0 then
        f:=1+f(n div 10)
   else f:=f(n div 10)
 else f:=0
end;
```

```
int p(long n)
{if (n)
  if(n%2==0)
       return 1+p(n/10);
  else return p(n/10);
 else return 0;
}
```

a. 0 **b.** 7 **c.** 3 **d.** 42

45. Se consideră subprogramul recursiv cu definiţia următoare. În urma apelului **p(n,2)** funcţia returnează valoarea **1** dacă şi numai dacă:

```
function p(n,x:integer):integer;

begin
 if(n<x) then p:=0;
  if (x=n) then p:=1
    else if (n mod x=0)
           then p:=0
           else
            p:=p(n,x+1)
end;
```

```
int p( unsigned int n,
       unsigned int x)
{
if (n<x) return 0;
 if (x==n) return 1;
     else if (n%x==0)
              return 0;
          else
         return p(n,x+1);
}
```

a. **n** este un număr natural impar **b.** **n** este un număr natural neprim
c. **n** este un număr natural par **d.** **n** este un număr natural prim

46. Care este valoarea returnată de funcţia cu definiţia următoare în urma apelului **f(4)**?

```
function f(x:integer):integer;
begin
  if (x<=0) then
        f:=3
           else
        f:=f(x-1)*2
end;
```

```
int f(int x)
{
if (x<=0)
  return 3;
   else
  return f(x-1)*2;
}
```

a. 48 **b.** 16 **c.** 24 **d.** 3

47. Se consideră subprogramul recursiv cu definiţia următoare:

```
Procedure p(n:integer);
begin
if (n<>1) then
 begin
  write (n,' ');
  if (n mod 2=0)
   then p(n div 2)
   else p(3*n+1)
 end
    else write (1)
end;
```

```
void p(int n)
{ if(n!=1)
  {
   printf("%d ",n);
          /cout<<n<<" ";
   if(n%2==0) p(n/2);
       else p(3*n+1);
  }
   else
printf("%d",1);/cout<<1;
}
```

Ce valori vor fi afişate pe ecran în urma apelului **p(10)**?

a. 5 16 8 4 2 1 **b.** 10 5 16 8 4 2 1
c. 10 5 16 8 4 2 **d.** 10 5 4 2 1

48. Se consideră subprogramul recursiv cu definiţia următoare. Ce valori vor fi afişate pe ecran în urma apelului **g(4)**?

```
procedure g(n:integer);
var i:integer;
begin
 if (n>=1) then
   begin
   for i:=1 to n do
    write (n,' ');
   g(n-1);
    end
end;
```

```
void g(int n)
{
 int i;
 if(n>=1)
 {for(i=1;i<=n;i++)
  printf("%d ",n);
       /cout<<n<<" ";
  g(n-1);
  }
}
```

a. 4 4 4 4 3 3 3 2 2 1 **b.** 4 1 2 3 4 4 3 2 1 4
c. 1 1 1 1 2 2 2 3 3 4 **d.** 4 3 2 1 3 2 1 2 1 1

49. Să se stabilească ce se va afişa pe ecran în urma executării apelului **Scrie(4)** pentru subprogramul definit astfel:

```
procedure Scrie(x: integer);
begin
  if (x > 0) then
   begin
     if x mod 2 = 1
        then writeln('###');
     Scrie(x - 1);
     if not(x mod 2=1)
         then writeln('***')
   end
end;
```

```
void Scrie (int x)
{if (x > 0)
    {if(x%2)
          printf("###\n");
          |cout << "###\n";
     Scrie(x - 1);
     if(!(x%2))
          printf("***\n");
          |cout << "***\n";
    }
}
```

a.
```
###
###
***
***
```
b.
```
***
###
***
###
```
c.
```
***
***
###
###
```
d.
```
###
***
###
***
```

50. Se consideră subprogramul **f** cu definiţia următoare şi o variabilă **y** de tip **integer /int**:

```
procedure f(x:integer; var k:integer);
  begin
   if x>1 then
       begin k:=k+1; f(x-1,k) end
   end;
```

```
void f(int x, int&k)
  {if (x>1)
          {k=k+1;
           f(x-1,k);}
   }
```

Ce se va afişa în urma executării secvenţei de program următoare:

```
y:=0; f(4, y);
write(y);
```

```
y=0; f(4, y);
cout<<y;/ printf("%d", y);
```

a. 4 **b.** 0 **c.** 5 **d.** 3

51. Ce valoare va returna **f(23951)**, pentru funcţia **f** definită mai jos?

```
function f(n:integer):integer;
var c:integer;
begin
if n=0 then f:=0
else begin
  c:= f(n div 10);
  if n mod 10>c then
       f:=n mod 10
  else f:=c
 end
end
```

```
int f(int n){
if (n==0) return 0;
else
{int c= f(n/10);
 if (n%10>c)
  return n%10;
 else return c;
}}
```

a. 2 **b.** 3 **c.** 5 **d.** 9

52. Ce valoare trebuie transmisă prin intermediul parametrului **n** la apelul funcţiei **f** definită mai jos, astfel încât valoarea returnată de funcţie să fie **7**?

```
function f(n:longint):integer;
begin
  if n=0 then f:=0
  else
    if n div 10 mod 2=1 then
      f:=n mod 10+f(n div 10)
    else f:=f(n div 10)
end;
```

```
int f(long n)
{ if (n==0) return 0;
  else
    if ((n/10)%2==1)
      return n%10+f(n/10);
    else return f(n/10);
}
```

a. 3258 **b.** 1528 **c.** 3972 **d.** 3472

53. Pentru funcţiile **f1** şi **f2** definite mai jos, stabiliţi care este rezultatul returnat la apelul **f2(305)**?

```
function f1(c:integer):longint;
begin
  if c<2 then f1:=1
  else f1:=c*f1(c-1)
end;

function f2(n:longint):longint;
begin
  if n=0 then f2:=0
  else f2:=f1(n%10)+f2(n/10)
end;
```

```
long f1(int c)
{if (c<1) return 1;
 else return c*f1(c-1);
}

long f2(long n)
{if (n==0) return 0;
 else
  return f1(n%10)+f2(n/10);
}
```

a. 126 **b.** 3 **c.** 127 **d.** 8

54. Pentru funcţiile **f1** şi **f2** definite mai jos, stabiliţi care este rezultatul returnat la apelul **f1(6)**?

```
function f2(x:longint):longint;
begin
 if x mod 2=0 then f2:=f2(x div 2)
 else f2:=x
end;
function f1(c:integer):longint;
begin
 if c<2 then f1:=1
 else f1:=f2(c*f1(c-1))
end;
```

```
long f2(long x)
{ if (x%2==0)
    return f2(x/2);
  else return x;
}

long f1(int c)
{ if (c<1) return 1;
  else
    return f2(c*f1(c-1));
}
```

a. 720 **b.** 16 **c.** 45 **d.** 360

55. Ce valoare va returna apelul **E(4)**?

```
function E(n:integer):integer;
begin
 if(n=0)or(n=1)then E:=1
 else E:=2*E(n-1)+E(n-2)
end;
```

```
int E(int n)
{
 if(n==0 || n==1) return 1;
 else
  return 2*E(n-1)+E(n-2);
}
```

a. 17 **b.** 15 **c.** 21 **d.** 9

56. Se consideră subprogramul **F** definit mai jos. De câte ori se afişează valoarea **0** dacă se apelează **F(9)**?

```
function F(x:integer):integer;
begin
  write(0);
  if x mod 2=0 then F:=x div 2
  else F:=F(F(x-1))
end;
```

```
int F(int x)
{
  cout<<0;  |  printf("0");
  if(x%2==0) return x/2;
  else return F(F(x-1));
}
```

a. 9 **b.** 2 **c.** 3 **d.** 12

57. Se consideră subprogramul **numar** având definiţia de mai jos. Care este valoarea returnată de funcţie pentru apelul **numar(5)**?

```
function numar(n:integer);
begin
 if n<=0 then numar:=0;
     else numar:=2*n+numar(n-1)
end;
```

```
int numar(int n)
{
  if (n<=0) return 0;
  else
    return 2*n+numar(n-1);
}
```

a. 15 **b.** 30 **c.** 10 **d.** 20

58. Care dintre următoarele subprograme afişează în ordine inversă (începând cu cifra unităţilor) cifrele unui număr natural, primit ca parametru?

a.
```
procedure numar(a:integer);
begin
  if a>9 then numar(a mod 10);
  write(a div 10)
end;
```

b.
```
procedure numar(a:integer);
begin
  write(a mod 10);
  if a>9 then numar(a div 10)
end;
```

c.
```
procedure numar(a:integer);
begin
  write(a div 10);
  if a>9 then numar(a mod 10)
end;
```

d.
```
procedure numar(a:integer);
begin
  if a>9 then numar(a div 10);
  write(a mod 10)
end;
```

a.
```
void numar(int a)
{if(a>9)numar(a%10);
cout<<a/10;
  /printf("%d",a/10);}
```

b.
```
void numar(int a)
{cout<<a%10;
   /  printf("%d",a%10);
if(a>9)numar(a/10);}
```

c.
```
void numar(int a)
{cout<<a/10;
   /  printf("%d",a/10);
if(a>9)numar(a%10);}
```

d.
```
void numar(int a)
{if(a>9)numar(a/10);
cout<<a%10;
   /
      printf("%d",a%10);
}
```

59. Se consideră subprogramul **f** având definiţia de mai jos. Care este valoarea returnată de subprogramul **f** după apelul: **f(10,2)**?

```
function f(x,y:integer):integer;
begin
 if x*y<=0 then f:=0
 else f:=1 + f(x-y,y)
end;
```

```
int f(int x, int y)
{ if (x*y<=0)return 0;
  else
    return 1 + f(x-y,y);
}
```

a. 3 **b.** 5 **c.** 1 **d.** 4

60. Fie funcţia **numara** prezentată mai jos. Care este apelul corect al funcţiei **numara** pentru a verifica dacă un număr natural **n** este prim?

```
function
numara(x,y:integer):integer;
begin
if y=0 then numara:=0
else
 if x mod y=0 then
  numara:=numara(x,y-1)+1
 else numara:=numara(x,y-1);
end;
```

a.
```
if numara(n,n)=2 then
   write('prim');
```

```
int numara(int x,int y)
{if (y==0) return 0;
 else
  if (x%y==0)
   return numara(x,y-1)+1;
  else
   return numara(x,y-1);
}
```

a.
```
if (numara(n,n)==2)
    cout<<"prim";
     /   printf("prim");
```

b.
```
if numara(2,n)=2 then
   write('prim');
```

c.
```
if numara(n,2)=0 then
    write('prim');
```

d.
```
if numara(n,n/2)=2 then
   write('prim');
```

b.
```
if (numara(2,n)==2)
  cout<<"prim";
  / printf("prim");
```

c.
```
if (numara(n,2)==0)
  cout<<"prim";
  / printf("prim");
```

d.
```
if (numara(n,n/2)==2)
  cout<<"prim";
  / printf("prim");
```

61. Pentru subprogramul de mai jos, apelul **`func(4)`** are ca rezultat:

```
function  func(x: integer):integer;
begin
 if x<=0 then func := -1
 else
func:=1 div func(x-1)-2*func(x-2)
end;
```

```
int func(int x) {
 if(x<=0) return -1;
 else
  return
  1/func(x-1)-2*func(x-2);
}
```

a. -1 **b.** -5 **c.** -6 **d.** 0

62. Fie subprogramul cu definiţia următoare. Care dintre următoarele instrucţiuni determină afişarea valorii **-7**?

```
function f (n:integer):integer;
begin
 if n<=0 then f:=-1
 else
  f:=f(n-1)-2*f(n-2)+3*f(n-3)
end;
```

```
int f (int n)
{if(n<=0) return -1;
 else
  return f(n-1)-2*f(n-2)+
                3*f(n-3);
}
```

a.
```
writeln(f(3))
```

b.
```
writeln(f(6))
```

c.
```
writeln(f(5))
```

d.
```
writeln(f(4))
```

a.
```
printf("%d",f(3)) | cout<<f(3)
```

b.
```
printf("%d",f(6)) | cout<<f(6)
```

c.
```
printf("%d",f(5)) | cout<<f(5)
```

d.
```
printf("%d",f(4)) | cout<<f(4)
```

63. Fie subprogramul cu definiţia următoare. Care dintre următoarele instrucţiuni determină afişarea valorii **-9**?

```
function f (n:integer):integer;
begin
 if n<=0 then f:=-1
 else
  f:=f(n-1)-2*f(n-2)+3*f(n-3)
end;
```

```
int f (int n)
{
 if(n<=0) return -1;
 else
  return f(n-1)-
         2*f(n-2)+3*f(n-3);}
```

a.
```
writeln(f(4))
```

b.
```
writeln(f(6))
```

c.
```
writeln(f(5))
```

d.
```
writeln(f(7))
```

a.
```
printf("%d",f(4)) | cout<<f(4)
```

b.
```
printf("%d",f(6)) | cout<<f(6)
```

c.
```
printf("%d",f(5)) | cout<<f(5)
```

d.
```
printf("%d",f(7)) | cout<<f(7)
```

64. Care trebuie să fie valoarea variabilei întregi **v** în apelul **F(v,s)** dacă valoarea iniţială a variabilei întregi **s** este **0**, iar valoarea acesteia în urma executării subprogramului **F**, definit mai jos, este **4301**?

```
procedure F( n:longint;
             var s :longint);
begin
 if n > 0 then
  begin
   s := s * 10 + n mod 10;
   F(n div 10, s)
  end
end;
```

```
void F(long int n,long int &s)
{ if (n > 0)
       {s = s * 10 + n % 10;
        F(n / 10, s);
       }
}
```

a. 1043 **b.** 1304 **c.** 4301 **d.** 1034

65. Ce valoare va avea variabila întreagă **s**, în urma apelului **Suma(1, s)**, ştiind că variabila globală **n** are valoarea **10**, variabila **s** are înainte de apel valoarea **0**, iar subprogramul **Suma** este definit mai jos?

```
procedure Suma ( i:integer;
                 var s:integer);
begin
 if i <= n then
   begin
    if i mod 2=1
            then s:=s+2
            else s := s - 1;
    Suma(i + 1, s)
   end
end;
```

```
void Suma(int i, int &s)
{ if (i <= n)
   {if(i % 2) s = s + 2;
         else  s = s - 1;
    Suma(i + 1, s);
   }
}
```

a. 8 **b.** 5 **c.** 7 **d.** 6

66. În subprogramul recursiv următor se consideră că **v** este un vector declarat global, format din numere întregi. Pentru care dintre vectorii **v** următori, expresia Pascal: **ddd(1,19)**, respectiv C/C++: **ddd(0,18)**, returnează valoarea **10**?

```
function ddd(i,j:byte):real;
begin
 if v[i]=v[j] then ddd:=i
 else ddd:=ddd(i+1,j-1)
end;
```

```
float ddd(int i,int j)
{
 if (v[i]==v[j]) return i;
 else return ddd(i+1,j-1);
}
```

a. (1,2,1,2,3,1,2,3,4,1,1,2,3,4,1,2,3,1,2)

b. (1,0,1,0,1,0,1,0,1,1,1,1,0,1,0,1,0,1,0)

c. (1,2,1,2,3,1,2,3,4,0,4,3,2,1,3,2,1,2,1)

d. (1,1,1,1,1,1,1,1,1,0,2,2,2,2,2,2,2,2,2)

67. Pentru un vector **x** declarat global, vector cu **10** componente întregi, stabiliţi care sunt valorile memorate de componentele tabloului **x** în urma apelului **ex(1,10)** (Pascal), respectiv, **ex(0,9)** (C/C++).

```
procedure ex(i,j:byte);
begin
   if i<=j then begin
      x[i]:=i;x[j]:=j;
      ex(i+1,j-1);
      if i mod 2=0 then begin
        x[i]:=j;
        x[j]:=11-x[i] end
     end
end;
```

```
void ex(int i,int j)
{if(i<=j)
 { x[i]=i;x[j]=j; ex(i+1,j-1);
   if(i%2!=0)
      {x[i]=j; x[j]=9-x[i];}
 }
}
```

a. (1,2,3,4,5,6,7,8,9,10)
b. (0,8,2,6,4,5,3,7,1,9)
c. (1,9,3,7,5,6,4,8,2,10)
d. (10,2,8,4,6,5,7,3,9,1)

a. (0,1,2,3,4,5,6,7,8,9)
b. (1,9,3,7,5,6,4,8,2,10)
c. (0,8,2,6,4,5,3,7,1,9)
d. (9,1,7,3,5,4,6,2,8,0)

68. Care este rezultatul afişat la apelul **f('examenul',1)**?

```
procedure f(s:string; i:byte);
begin
  if i<=length(s)
   then begin
    f(s,i+1);
    if pos(s[i],'aeiou')<>0
        then write(s[i])
  end
end;
```

```
void f(char s[256],int i)
{if (i<strlen(s))
  {f(s,i+1);
   if(strchr("aeiou",
       s[i])!=0)
     cout<<s[i];
    |printf("%c",s[i]);
  }
}
```

a. eaeu **b.** examenul **c.** ueae **d.** e

69. Fie vectorul **v** cu indicii de la **0** la **7**, vector ce reţine, în ordine, următoarele valori întregi: **100,200,243,132,413,352,222,341**. Care este apelul corect al subprogramului următor astfel încât valoarea returnată să fie **5**?

```
function f(poz,c:integer;
           var v:vector): integer;
var nr:integer;
begin
 if poz<0 then f:=c
 else begin
   nr:=v[poz];
   while nr>0 do begin
     if nr mod 10>c then
        c:=nr mod 10;
     nr:=nr div 10
   end;
   f:=f(poz-1,c,v) end
end;
```

```
int f(int poz,int v[],int c)
{  if(poz<0) return c;
   else {
     int nr=v[poz];
     while(nr) {
       if(nr%10>c) c=nr%10;
       nr=nr/10;
     }
     return f(poz-1,v,c);
   }
}
```

a. `writeln(f(4,4,v))`	**a.** `printf("%d",f(4,v,4)) / cout<<f(4,v,4)`
b. `writeln(f(7,0,v))`	**b.** `printf("%d",f(7,v,0)) / cout<<f(7,v,0)`
c. `writeln(f(7,6,v))`	**c.** `printf("%d",f(7,v,6)) / cout<<f(7,v,6)`
d. `writeln(f(4,0,v))`	**d.** `printf("%d",f(4,v,0)) / cout<<f(4,v,0)`

70. Fie vectorul **v** cu indicii de la **0** la **7**, vector ce reţine, în ordine, următoarele valori întregi: **88, 777,789,976,998,7856,7887,8979**
Care este apelul corect al subprogramului următor astfel încât valoarea returnată să fie **5**?

```
function f(poz,c:integer;
       var v:vector):integer;
var nr:integer;
begin
   if poz<0 then f:=c
   else begin nr:=v[poz];
      while nr>0 do begin
        if nr mod 10 < c then
          c:=nr mod 10;
        nr:=nr div 10
      end;
      f:=f(poz-1,c,v)
    end
end;
```

```
int f(int poz,int v[],int c)
{ if(poz<0) return c;
  else
  { int nr=v[poz];
    while(nr) {
       if(nr%10<c) c=nr%10;
       nr=nr/10; }
    return f(poz-1,v,c);
  }
}
```

a. `writeln(f(5,9,v))`	**a.** `printf("%d",f(5,v,9)) | cout<<f(5,v,9)`
b. `writeln(f(5,0,v))`	**b.** `printf("%d",f(5,v,0)) | cout<<f(5,v,0)`
c. `writeln(f(4,9,v))`	**c.** `printf("%d",f(4,v,9)) | cout<<f(4,v,9)`
d. `writeln(f(3,9,v))`	**d.** `printf("%d",f(3,v,9)) | cout<<f(3,v,9)`

71. Ştiind că **p** este un vector cu **3** componente întregi (vector declarat global), **M** este mulţimea tuturor cifrelor nenule, iar subprogramul **tipar** afişează valorile elementelor

p[1],p[2] şi p[3],	**p[0],p[1] şi p[2],**

cu ce trebuie înlocuite simbolurile **α**, **β** şi **γ** în definiţia funcţiei **G** astfel încât în urma apelului **G(1)** să se afişeze toate elementele produsului cartezian **M×M×M**?

```
procedure G(k:integer);
var i:integer;
begin
for i:=α to β do begin
  p[k]:=i;
  if k=γ then tipar
  else G(k+1)
  end
end;
```

```
void G(int k)
{int i;
for(i=α;i<=β;i++){
  p[k]=i ;
  if(k==γ)tipar();
  else G(k+1);
 }
}
```

a. α=0 β=10 γ=3	**a.** α=0 β=10 γ=3
b. α=1 β=3 γ=9	**b.** α=1 β=3 γ=9
c. α=1 β=9 γ=2	**c.** α=1 β=9 γ=3
d. α=1 β=9 γ=3	**d.** α=1 β=9 γ=2

4.2.2. Probleme (programe pseudocod, Pascal | C / C++)

1. Se consideră programul pseudocod alăturat:

S-a notat cu **[x]** partea întreagă a numărului real **x**.

1) Care sunt valorile afişate pentru **n=2456753**?

```
citeşte n(număr natural)
a←0
┌cât timp n%2≠0execută
│ a←a*10+n%10
│ n←[n/10]
└■
scrie n,a
```

2) Stabiliţi o valoare de trei cifre pentru variabila **n** astfel încât, după executarea programului, **a** să conţină inversul (definit ca numărul obţinut din cifrele numărului iniţial, aşezate exact în ordine inversă) valorii iniţiale a lui **n**.

3) Scrieţi programul **PASCAL/C/C++** corespunzător algoritmului dat.

4) Scrieţi în limbajul **PASCAL/C/C++** un subprogram recursiv **sub**, având un singur parametru, şi care, la apelul **sub(n)**, să afişeze aceleaşi rezultate ca şi programul dat, pentru orice valoare naturală a lui **n**.

2. Scrieţi un subprogram recursiv cu un singur parametru **n**, număr natural cu cel mult **9** cifre, care să afişeze pe ecran numai cifrele impare ale lui **n**, în ordinea în care ele apar în număr, ca în exemplul de mai jos. Dacă numărul are numai cifre pare, nu se va afişa nimic.
De exemplu, pentru **n=29385567** se va afişa **93557**.

3. Scrieţi un program Pascal/C/C++ care citeşte de la tastatură un număr natural **n (0<n<100)**. Să se creeze fişierul **BAC.TXT** în care să se scrie primii **n** termeni ai şirului Fibonacci în ordine descrescătoare separaţi printr-un spaţiu.
Se defineşte şirul Fibonacci astfel : $f_1=1;\ f_2=1;\ \ldots\ ;f_n = f_{n-1} + f_{n-2}$
Exemplu: pentru **n=8** fişierul **BAC.TXT** va conţine şirul :
21 13 8 5 3 2 1 1

4. Realizaţi următoarele cerinţe utilizând limbajul Pascal/C/C++:
a) Scrieţi definiţia completă a unui subprogram recursiv **sub**, care primeşte prin intermediul parametrului **n** un număr natural cu cel mult **8** cifre şi returnează prin intermediul celui de-al doilea parametru **max**, cea mai mare cifră din scrierea lui **n**;
b) Scrieţi un program care citeşte de la tastatură un număr natural **nr** cu cel mult **8** cifre şi care, folosind subprogramul **sub**, afişează pe ecran cea mai mare cifră a numărului **nr** şi de câte ori apare aceasta în scrierea sa.
Exemplu, pentru **nr=26361**, se afişează: **6 2**.

5. Scrieţi un program Pascal/C/C++ care citeşte din fişierul text **DATE.IN,** cel mult **100** de numere naturale nenule aflate pe o singură linie, formate din cel mult patru cifre fiecare, separate prin spaţii şi scrie aceste numere în fisierul text **DATE.OUT**, în ordine inversă faţă de cea în care fost citite, pe o singură linie, separate prin spaţii.

De exemplu, dacă din fişierul **DATE.IN** se citesc numerele: **93 207 15 1892 3762**, atunci conţinutul fişierului **DATE.OUT** va fi: **3762 1892 15 207 93**

6. Realizaţi următoarele cerinţe utilizând limbajul Pascal/C/C++:

a) Scrieţi definiţia completă a unui subprogram recursiv sub care returnează **1** dacă un număr natural mai mic decât **30000,** dat prin intermediul parametrului nr, este prim şi returnează **0** în caz contrar.

b) Scrieţi un program care citeşte din fişierul **bac.txt** un număr natural **n** (**n<1000**) şi un şir de n numere naturale mai mici decât **30000**, separate prin caracterul spaţiu; programul determină şi afişează pe ecran secvenţa de lungime maximă de termeni ai şirului, aflaţi pe poziţii consecutive şi care sunt numere prime, separaţi prin câte un caracter spaţiu, folosind apeluri ale subprogramului de la punctul a.

Exemplu: Dacă fişierul **bac.txt** conţine valorile: **8 4 7 3 6 11 17 101 45** se afişează **11 17 101**

5. Combinatorică şi tehnica Backtracking

5.1. Teste grilă

1. Se generează toate numerele naturale de 4 cifre, cifre aflate în ordine strict crescătoare, orice două cifre vecine din fiecare număr generat fiind valori neconsecutive. De exemplu, numerele **1579** şi **2468** sunt în şirul numerelor generate, în timp ce **3851**, **1679**, **479** nu sunt. Câte numere se generează în total?

a. 12 **b.** 15 **c.** 20 **d.** 24

2. Folosind modelul combinărilor, se generează cuvinte cu câte două litere distincte din mulţimea **{i,t,e,m}** obţinându-se, în ordine: **it, ie, im, te, tm, em**. Dacă se utilizează exact aceeaşi tehnică pentru a genera cuvinte cu trei litere distincte din mulţimea **{a,i,t,e,m}**, atunci antepenultimul cuvânt generat este:

a. iem **b.** itm **c.** atm **d.** tem

3. Folosind modelul combinărilor, se generează cuvinte cu câte două litere distincte din mulţimea **{i,t,e,m}** obţinându-se, în ordine: **it, ie, im, te, tm, em**. Dacă se utilizează exact aceeaşi tehnică pentru a genera cuvinte cu patru litere distincte din mulţimea **{i,t,e,m,a,x}**, atunci numărul de cuvinte generate care încep cu litera **t** este:

a. 24 **b.** 12 **c.** 16 **d.** 4

4. Folosind modelul combinărilor se generează cuvinte cu câte două litere distincte din mulţimea **{i,t,e,m}** obţinându-se, în ordine: **it, ie, im, te, tm, em**. Dacă se utilizează exact aceeaşi tehnică pentru a genera toate cuvintele cu patru litere distincte din mulţimea **{i,t,e,m,a,x}**, atunci predecesorul şi succesorul cuvântului **tema** generat la un moment dat sunt, în această ordine:

a. iemx temx
b. imax teax
c. imax temx
d. item emax

5. Folosind modelul combinărilor se generează cuvinte cu câte două litere distincte din mulţimea **{i,t,e,m}** obţinându-se, în ordine: **it, ie, im, te, tm, em**. Dacă se utilizează exact aceeaşi tehnică pentru a genera cuvinte cu patru litere distincte din mulţimea **{i,t,e,m,a,x}**, atunci numărul de cuvinte generate care se termină cu litera **a** este:

a. 4 **b.** 12 **c.** 24 **d.** 5

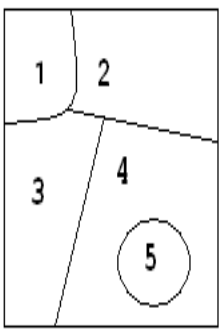
6. Folosind modelul combinărilor se generează cuvinte cu câte trei litere distincte din mulţimea **{i,t,e,m}** obţinându-se, în ordine: **ite, itm, iem, tem**. Dacă se utilizează exact aceeaşi tehnică pentru a genera cuvinte cu patru litere distincte din mulţimea **{c,r,i,t,e,m,a,s}**, atunci numărul de cuvinte generate care încep cu litera **r** şi se termină cu litera **a** sau cu litera **s** este:

a. 30 **b.** 20 **c.** 16 **d.** 12

7. Se consideră mulţimea **{4, 1, 2, 3}**. Dacă se generează toate permutările elementelor acestei mulţimi, în câte dintre acestea elementele **1** şi **2** apar pe poziţii consecutive, **în această ordine** (ca în permutările **(1,2,3,4)** sau **(3,1,2,4)**)?

a. 8 **b.** 24 **c.** 6 **d.** 12

8. Desenul alăturat reprezintă o hartă cu **5** ţări numerotate de la **1** la **5**. Se generează toate variantele de colorare a acestei hărţi având la dispoziţie **4** culori notate cu **A, B, C, D**, astfel încât oricare două ţări vecine să nu fie colorate la fel.



Prima soluţie este **(A,B,C,A,B)** având următoarea semnificaţie: ţara **1** e colorată cu **A**, ţara **2** e colorată cu **B**, ţara **3** e colorată cu **C**, ţara **4** e colorată cu **A**, ţara **5** e colorată cu **B**.

Ştiind că următoarele trei soluţii sunt obţinute în ordinea

(A,B,C,A,C), (A,B,C,A,D), (A,B,C,D,A),

care este soluţia care se obţine după varianta de colorare **(C,A,B,D,C)**?

a. (D,A,B,D,A) **b.** (C,A,D,B,A) **c.** (C,D,B,A,B) **d.** (C,A,B,C,D)

9. Se generează toate numerele de **5** cifre, cu cifre distincte, care pe poziţii pare au cifre pare, iar pe poziţii impare au cifre impare. Primele şase numere generate sunt:

10325, **10327**, **10329**, **10345**, **10347**, **10349**.

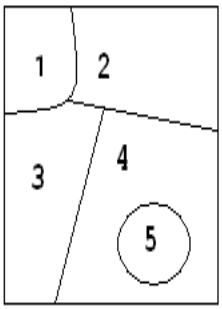
Care este următorul număr generat după numărul **96785**?

a. 96587 **b.** 98123 **c.** 96783 **d.** 98103

10. Se generează produsul cartezian al mulţimilor **{1,2,3}**, **{1,2}**, **{3,4,5}**. Câte dintre elementele produsului cartezian conţin cel puţin o valoare egală cu **1**?

a. 18 **b.** 6 **c.** 24 **d.** 12

11. Desenul alăturat reprezintă o hartă cu **5** ţări numerotate de la **1** la **5**. Se generează toate variantele de colorare a acestei hărţi având la dispoziţie **4** culori notate cu **A, B, C, D**, astfel încât oricare două ţări vecine să nu fie colorate la fel. Prima soluţie este **(A, B, C, A, B)** având următoarea semnificaţie: ţara **1** e colorată cu **A**, ţara **2** e colorată cu **B**, ţara **3** e colorată cu **C**, ţara **4** e colorată cu **A**, ţara **5** e colorată cu **B**. Care din următoarele variante poate reprezenta o soluţie de colorare?



a. (C,D,B,A,A) **b.** (D,B,D,A,C) **c.** (D,C,B,D,C) **d.** (C,B,D,B,A)

12. Se generează matricele pătratice cu **n** linii şi **n** coloane cu elemente **0** şi **1** care pe fiecare linie au un singur element egal cu **1**, pe fiecare coloană au un singur element egal cu **1**, iar restul elementelor sunt nule. Dacă **n=3**, matricele sunt generate în ordinea următoare:

```
100   100   010   010   001   001
010   001   100   001   100   010
001   010   001   100   010   100
```

Dacă **n=4**, care este matricea generată imediat după matricea:

```
0010
1000
0001
0100
```

a.
```
0010
1000
0100
0001
```
b.
```
0010
0100
1000
0001
```
c.
```
0001
1000
0010
0100
```
d.
```
0010
0001
1000
0100
```

13. Generarea tuturor şirurilor de **4** elemente, fiecare element putând fi orice literă din mulţimea **{a,b,m,k,o,t}**, se realizează cu ajutorul unui algoritm echivalent cu algoritmul de generare a:

a. produsului cartezian **b.** permutărilor

c. aranjamentelor **d.** combinărilor

14. Folosind primele patru numere prime, se construiesc, în ordine, următoarele sume: **2; 2+3; 2+3+5; 2+3+5+7; 2+3+7; 2+5; 2+5+7; 2+7; 3; 3+5; 3+5+7; 3+7; 5; 5+7; 7**. Folosind aceeaşi metodă, construim sume utilizând primele cinci numere prime. Care este a şasea sumă, astfel obţinută?

a. 2+3+5+11 **b.** 2+3+7 **c.** 3+5+11 **d.** 2+3+5+7+11

15. Folosind **metoda backtracking**, se construiesc numere cu cifre distincte, numere care au suma cifrelor egală cu **5** şi nu sunt divizibile cu **10**. Se obţin, în acestă ordine, numerele: **104; 14; 203; 23; 302; 32; 401; 41; 5.** Care este **al şaselea număr** obţinut dacă, folosind acelaşi algoritm, se construiesc numere naturale cu cifre diferite, nedivizibile cu **10** şi cu suma cifrelor egală cu **6**.

a. 213 **b.** 1302 **c.** 2013 **d.** 15

16. Folosind numai cifrele **{0,5,3,8}**, se construiesc, prin metoda backtracking, toate numerele cu **3** cifre în care oricare două cifre alăturate nu au aceeaşi paritate. Se obţin, în ordine numerele: **505, 503, 585, 583, 305, 303, 385, 383, 850, 858, 830, 838**. Utilizând acelaşi algoritm pentru a obţine numere cu patru cifre din mulţimea **{0,3,6,2,9}**, în care oricare două cifre alăturate nu au aceeaşi paritate, al şaselea număr care se obţine este:

a. 3092 **b.** 3690 **c.** 6309 **d.** 3096

17. Un elev, folosind **metoda backtracking**, construieşte toate numerele cu cifre distincte, numere care au suma cifrelor egală cu **5** şi nu sunt divizibile cu **10**. El obţine, în această ordine, numerele: **104; 14; 203; 23; 302; 32; 401; 41; 5.** Folosind aceeaşi metodă, el construieşte toate numerele naturale cu cifre diferite, nedivizibile cu **10** şi cu suma cifrelor egală cu **6.** Care sunt **primele patru numere** pe care le construieşte?

a. 1023; 105; 15; 6 **b.** 123; 132; 15; 213

c. 1023; 123; 1032; 132 **d.** 1023; 1032; 105; 1203;

18. Folosind cifrele **{0,5,3,8}**, se generează toate numerele cu **3** cifre cu proprietatea că oricare două cifre alăturate nu au aceeaşi paritate. Astfel, se obţin în ordine numerele: **505, 503, 585, 583, 305, 303, 385, 383, 850, 858, 830,838**. Folosind aceeaşi metodă, se generează numere de patru cifre din mulţimea **{0,3,6,2,9}**, ultimul număr astfel obţinut este:

a. 9292 **b.** 3629 **c.** 9692 **d.** 9632

19. Pentru **n=4151**, stabiliţi câte numere strict mai mari decât **n** şi având exact aceleaşi cifre ca şi **n** există.

a. 5 **b.** 4 **c.** 2 **d.** 3

20. Se generează toate şirurile **6** de paranteze care se închid corect: **()(())**, **((()))**, **(())()**, **()()()**. Lipseşte vreo soluţie?

a. Da, trei soluţii **b.** Da, una singură

c. Nu **d.** Da, două soluţii

21. Problema generării tuturor numerelor de **n** cifre **(n≤9)** cu cifrele în ordine strict crescătoare este similară cu problema:

a. generării permutărilor de **n** elemente
b. generării combinărilor de **9** elemente luate câte **n**
c. generării combinărilor de **n** elemente luate câte **9**
d. generării aranjamentelor de **9** elemente luate câte **n**

22. Pentru a scrie valoarea **10** ca sumă de numere prime se foloseşte metoda backtracking şi se generează, în această ordine, sumele distincte: **2+2+2+2+2**, **2+2+3+3**, **2+3+5**, **3+7**, **5+5**. Folosind exact aceeaşi metodă, se scrie valoarea **9** ca sumă de numere prime.

Care este a doua soluţie?

a. **2+2+2+3** **b.** **2+2+5** **c.** **2+2+3+2** **d.** **2+7**

23. Un program foloseşte metoda backtracking pentru a afişa toate steagurile tricolore formate cu culorile **alb**, **albastru**, **galben**, **mov**, **negru**, **portocaliu**, **roşu**, **verde**. Se ştie că în mijloc singurele culori care pot fi folosite sunt **alb, galben** sau **portocaliu**, iar cele trei culori dintr-un steag trebuie să fie distincte două câte două. Primele patru steaguri generate de program sunt: (**alb**, **galben**, **albastru**), (**alb**, **galben**, **mov**), (**alb**, **galben**, **negru**), (**alb**, **galben**, **portocaliu**).

Care este cel de al optulea steag generat de program?

a. **alb**, **portocaliu**, **mov** **b.** **alb**, **portocaliu**, **albastru**

c. **albastru**, **alb**, **galben** **d.** **alb**, **portocaliu**, **galben**

24. Trei băieţi **A**, **B** şi **C**, si trei fete **D**, **E** şi **F**, trebuie să formeze o echipă de trei copii, care să participe la un concurs. Echipa trebuie să fie mixtă (adică să conţină cel puţin o fată şi cel puţin un băiat). Ordinea copiilor în echipă este importantă deoarece aceasta va fi ordinea de intrare a copiilor în concurs (de exemplu echipa **A**, **B**, **D** este diferită de echipa **B**, **A**, **D**).

În câte dintre echipele formate se găsesc atât băiatul **A** cât şi băiatul **B**?

a. **3** **b.** **36** **c.** **18** **d.** **6**

25. Se dă o mulţime de **n** puncte în plan. Se ştie că oricare **3** dintre aceste puncte nu sunt coliniare. Se cere să se genereze toate triunghiurile având vârfurile în mulţimea dată. Cu ce algoritm este echivalent algoritmul de rezolvare a acestei probleme?

a. Generarea combinărilor de **n** elemente luate câte **3**
b. Generarea aranjamentelor de **n** elemente luate câte **3**
c. Generarea partiţiilor unei mulţimi cu **n** elemente.
d. Generarea tuturor submulţimilor unei mulţimi cu **n** elemente.

26. Un program folosind un algoritm backtracking generează, în ordine lexicografică, toate anagramele distincte ale cuvântului **babac**. Primele **5** anagrame generate de acest algoritm sunt **aabbc, aabcb, aacbb, ababc, abacb**.

Care este cea de a zecea anagramă generată de acest program?

a. **acbab** **b.** **acabb** **c.** **baabc** **d.** **abcba**

27. Un program generează în ordine lexicografică toate şirurile de **3** litere având următoarele proprietăţi: şirurile sunt formate doar din litere mari ale alfabetului englez, toate literele din şir sunt distincte, oricare două litere alăturate din şir sunt consecutive în alfabet.

Primele **6** şiruri generate de acest program sunt: **ABC**, **BCD**, **CBA**, **CDE**, **DCB**, **DEF**. Care este cea de a **noua** soluţie generată de acest program.

a. **FED** **b.** **FGH** **c.** **IJK** **d.** **LKJ**

28. Un algoritm de tip backtracking generează, în ordine lexicografică, toate şirurile de **5** cifre **0** şi **1** cu proprietatea că nu există mai mult de două cifre de **0** consecutive. Primele **6** soluţii generate sunt: **00100**, **00101**, **00110**, **00111**, **01001**, **01010**. Care este cea de a **opta** soluţie?

a. **01110** **b.** **01100** **c.** **01011** **d.** **01101**

29. Problema determinării tuturor modalităţilor de a-i împărţii pe cei **n** elevi ai unei clase în echipe, astfel încât fiecare elev să facă parte dintr-o echipă şi în fiecare echipă să fie minimum un elev şi maximum **n** elevi, este similară cu:

a. generarea tuturor submulţimilor unei mulţimi cu **n** elemente

b. generarea produsului cartezian a **n** mulţimi, cu câte **n** elemente fiecare

c. generarea tuturor partiţiilor unei mulţimi cu **n** elemente

d. generarea tuturor permutărilor de **n** elemente

30. Aplicând metoda backtracking pentru a genera toate permutările celor **n** elemente ale unei mulţimi, o soluţie se memorează sub forma unui tablou unidimensional $x_1, x_2 \ldots x_n$. Dacă sunt deja generate valori pentru componentele $x_1, x_2 \ldots x_{k-1}$, iar pentru componenta curentă, x_k **(1<k<n)**, au fost testate toate valorile posibile şi nu a fost găsită niciuna convenabilă, atunci:

a. se încearcă alegerea unei valori pentru componenta x_{k-1}

b. se încheie algoritmul

c. se încearcă alegerea unei valori pentru componenta x_1 oricare ar fi **k**

d. se încearcă alegerea unei valori pentru componenta x_{k+1}

31. Utilizăm metoda backtracking pentru a genera toate cuvintele alcătuite din două litere ale mulţimii **{a, c, e, g}**, astfel încât să nu existe două consoane alăturate. Cuvintele se generează în următoarea ordine: **aa, ac, ae, ag, ca, ce, ea, ec, ee, eg, ga, ge**. Dacă se utilizează exact aceeaşi metodă pentru a genera cuvintele formate din **4** litere ale mulţimii **{a, b, c, d, e, f}**, astfel încât să nu existe două consoane alăturate în cuvânt, care este penultimul cuvânt generat?

a. **fefa** **b.** **fafe** **c.** **feef** **d.** **fefe**

32. Utilizând metoda backtracking se generează toate numerele formate doar din **3** cifre astfel încât fiecare număr să aibă cifrele distincte. Cifrele fiecărui număr sunt din mulţimea **{1, 2, 3, 4}** . Acest algoritm generează numerele, în această ordine: **123, 124, 132, 134, 213, 214, 231, 234, 312, 314, 321, 324, 412, 413, 421, 423, 431, 432**. Dacă utilizăm acelaşi algoritm pentru a genera toate numerele de **4** cifre, fiecare număr fiind format din cifre distincte din mulţimea **{1**, **2**, **3**, **4**, **5}**, precizaţi care este numărul generat imediat după **4325**.

a. **4351** **b.** **5123** **c.** **4521** **d.** **4321**

33. Utilizând metoda backtracking se generează toate numerele palindrom formate din **4** cifre. Fiecare număr conţine cifre din mulţimea **{1, 3, 5}.** Elementele sunt generate în următoarea ordine: **1111, 1331, 1551, 3113, 3333, 3553, 5115, 5335, 5555**. Dacă se utilizează exact aceeaşi metodă pentru a genera toate numerele palindrom formate din **4** cifre, fiecare element având cifre din mulţimea **{1, 2, 3, 4, 5, 6, 7, 8, 9}**, să se precizeze câte numere pare se vor genera.

a. **99** **b.** **40** **c.** **36** **d.** **72**

34. Utilizând metoda backtracking se generează elementele produsului cartezian a **n** mulţimi: $A_1, A_2, \ldots, A_n$. Dacă utilizăm acest algoritm pentru a genera elementele produsului cartezian a **3** mulţimi: **M={1, 2, 3} N={1, 2} şi P={1, 2, 3, 4}** atunci care din următoarele secvenţe **nu** reprezintă o soluţie a acestui algoritm, pentru produsul cartezian **P×N×M**?

a. **(4,2,3)** **b.** **(3,3,3)** **c.** **(3,2,1)** **d.** **(1,1,1)**

35. Utilizând metoda backtracking se generează toate numerele de câte trei cifre astfel încât fiecare număr generat are cifrele distincte şi suma lor este un număr par. Precizaţi care dintre următoarele numere reprezintă o soluţie a algoritmului?

a. **235** **b.** **455** **c.** **986** **d.** **282**

36. Se generează prin metoda **backtracking** mulţimi distincte cu elemente numere naturale nenule şi cu proprietatea că suma elementelor fiecărei mulţimi este egală cu **7** astfel: {**1**, **2**, **4**}, {**1**, **6**}, {**2**, **5**}, {**3**, **4**}, {**7**}. Folosind aceeaşi metodă pentru a genera mulţimi distincte cu elemente numere naturale nenule şi cu proprietatea că suma elementelor fiecărei mulţimi este egală cu **9**, stabiliţi în ce ordine sunt generate următoarele mulţimi:
a) **{2, 3, 4};** **b)** **{3, 6};** **c)** **{2, 7};** **d)** **{1, 8}.**

a. **d a b c** **b.** **d a c b** **c.** **a c b d** **d.** **a b c d**

37. Se generează toate şirurile strict crescătoare de numere naturale nenule mai mici sau egale cu **4**, având primul termen **1** sau **2**, ultimul termen **4** şi cu diferenţa dintre oricare doi termeni aflaţi pe poziţii consecutive cel mult **2**, obţinându-se soluţiile: (**1,2,3,4**), (**1,2,4**), (**1,3,4**), (**2,3,4**), (**2,4**). Folosind aceeaşi metodă, generăm toate şirurile strict crescătoare de numere naturale nenule mai mici sau egale cu **5**, care dintre afirmaţiile următoare este adevărată:

a. imediat după soluţia (**1,3,5**) se generează soluţia (**2,3,4,5**)

b. imediat după soluţia (**2,3,5**) se generează (**2,5**)

c. penultima soluţie generată este (**2,4,5**)

d. în total sunt generate **5** soluţii

38. Se generează toate şirurile strict crescătoare de numere naturale nenule mai mici sau egale cu **4**, având primul termen **1** sau **2**, ultimul termen **4** şi cu diferenţa dintre oricare doi termeni aflaţi pe poziţii consecutive cel mult **2**, obţinându-se soluţiile: (**1,2,3,4**), (**1,2,4**), (**1,3,4**), (**2,3,4**), (**2,4**). Folosind aceeaşi metodă, generăm toate şirurile strict crescătoare de numere naturale nenule mai mici sau egale cu **6**, având primul termen **1** sau **2**, ultimul termen **6** şi diferenţa dintre oricare doi termeni aflaţi pe poziţii consecutive cel mult **2**, care dintre afirmaţiile următoare este adevărată?

a. imediat după soluţia (**1,3,4,5,6**) se generează soluţia (**2,3,4,5,6**);

b. penultima soluţie generată este (**2,3,5,6**);

c. imediat după soluţia (**1,2,4,6**) se generează soluţia (**1,3,4,6**);

d. în total sunt generate **13** soluţii;

39. Dirigintele unei clase trebuie să aleagă trei elevi pentru un concurs. Elevii respectivei clase i-au propus pe Ionel, Gigel, Dorel, şi Viorel. Pentru a decide dirigintele foloseşte un algoritm Backtracking care să îi genereze toate soluţiile posibile. Câte soluţii vor fi generate?

a. **12** **b.** **24** **c.** **6** **d.** **4**

40. Se generează toate şirurile strict crescătoare de numere naturale nenule mai mici sau egale cu **4**, având primul termen **1** sau **2**, ultimul termen **4** şi cu diferenţa dintre oricare doi termeni aflaţi pe poziţii consecutive cel mult **2**, obţinându-se soluţiile: (**1,2,3,4**), (**1,2,4**), (**1,3,4**), (**2,3,4**), (**2,4**). Folosind aceeaşi metodă, generăm toate şirurile strict crescătoare de numere naturale nenule mai mici sau egale cu **6**, având primul termen **1** sau **2**, ultimul termen **6** şi diferenţa dintre oricare doi termeni aflaţi pe poziţii consecutive cel mult **2**, care dintre afirmaţiile următoare este adevărată:

a. (**1,3,5,6**) nu este soluţie

b. a şasea soluţie generată este (**1,3,4,5,6**)

c. ultima soluţie generată este o mulţime cu **4** elemente

d. în total sunt generate cel mult **10** soluţii

41. Se generează în ordine crescătoare numerele de câte şase cifre care conţin: cifra `1` o singură dată, cifra `2` de două ori şi cifra `3` de trei ori. Se obţin, în această ordine, numerele: **122333**, **123233**, **123323**, …, **333221**. Care dintre următoarele propoziţii este adevărată?

a. imediat după numărul **332312** se generează **332321**

b. sunt **8** numere generate prin această metodă care au prima cifră **1** şi ultima cifră **2**

c. sunt **6** numere generate prin această metodă care au prima cifră **1** şi a doua cifră **2**

d. penultimul număr astfel generat este **333122**

42. Având la dispoziţie gama celor **7** note muzicale, algoritmul de generare a tuturor succesiunilor (melodiilor) distincte formate din exact **100** de note este similar cu algoritmul de generare a:

a. aranjamentelor **b.** partiţiilor unei mulţimi
c. permutărilor **d.** elementelor produsului cartezian

43. Se consideră mulţimea **{1,7,5,16,12}**; se generează prin metoda backtracking toate submulţimile sale formate din exact **3** elemente: primele patru soluţii generate sunt, în ordine: **{1,7,5}**, **{1,7,16}**, **{1,7,12}**, **{1,5,16}**. Care dintre soluţii trebuie eliminată din şirul următor astfel încât cele rămase să apară în şir în ordinea generării lor?
{1,5,12}, {5,16,12}, {7,5,16}, {7,5,12}

a. **{1,5,12}** **b.** **{7,5,16}** **c.** **{7,5,12}** **d.** **{5,16,12}**

44. Având la dispoziţie cifrele **0**, **1** şi **2** putem genera, în ordine crescătoare, numere care au suma cifrelor egală cu **2** astfel: **2, 11, 20, 101, 110, 200, etc**. Folosind acest algoritm generaţi numere cu cifrele **0**, **1** şi **2** care au suma cifrelor egală cu **3**. Care va fi al şaptelea număr din această generare ?

a. **120** **b.** **1002** **c.** **201** **d.** **210**

45. Cele **4** prietene Dana, Alina, Oana şi Maria doresc să stea împreună în clasă, într-o bancă cu **3** locuri. În câte modalităţi se pot aranja în bancă ştiind că unul dintre cele **3** locuri îl va ocupa întotdeauna Oana.

a. **36** **b.** **24** **c.** **18** **d.** **12**

46. Folosind un algoritm de generare putem obţine numere naturale de **k** cifre care au suma cifrelor egală cu un număr natural **s** introdus de la tastatură, unde **s** şi **k** sunt numere naturale nenule. Astfel pentru valorile **k=2** şi **s=6** se generează numerele: **15**, **24**, **33**, **42**, **51**, **60**. Care vor fi primele **4** numere ce se vor genera pentru **k=3** şi **s=8**?

a. **800, 710, 620, 530** **b.** **107, 116, 125, 134**
c. **125, 233, 341, 431** **d.** **116, 125, 134, 143**

47. Elevii unei clase trebuie să programeze **4** probe de evaluare la matematică, română, informatică şi istorie, pe parcursul a **8** zile de şcoală. În câte moduri pot realiza această programare, ştiind că nu este permisă programarea a două probe în aceeaşi zi?

a. 1680 **b.** 32 **c.** 1760 **d.** 24

48. Un număr este palindrom dacă citit de la stânga la dreapta sau invers reprezintă acelaşi număr. Generăm palindroamele de lungime **3** având la dispoziţie cifrele **0,1,2,3,4**, şi obţinem numerele: **101, 111, 121, 131, 141, 202, 212, 222**, etc. Folosind exact acelaşi procedeu, care este al şaptelea număr din generarea palindroamelor de lungime **4** având la dispoziţie cifrele **0,1,2,3,4,5**?

a. 5005 **b.** 2002 **c.** 1551 **d.** 2121

49. Generarea tuturor cuvintelor de **4** litere, fiecare literă putând fi orice element din mulţimea **{a,c,e,m,o,s},** se realizează cu ajutorul unui algoritm echivalent cu algoritmul de generare a:

a. produsului cartezian **c.** partiţiilor unei mulţimi
b. combinărilor **d.** permutărilor

50. Se consideră mulţimile **A={1,2,3} , B={1} , C={2,3,4}.** Elementele produsului cartezian **AxBxC** se generează, în ordine, astfel **(1,1,2), (1,1,3), (1,1,4), (2,1,2), (2,1,3), (2,1,4), (3,1,2), (3,1,3), (3,1,4).** Dacă, prin acelaşi algoritm se generează produsul cartezian al mulţimilor **AxBxC** unde **A={a}, B={a,b},C={b,c,d},** atunci cel de-al patrulea element generat este :

a. (a,b,c) **b.** (a,c,b) **c.** (a,b,b) **d.** (a,c,d)

51. Pentru a determina toate modalităţile de a scrie numărul **8** ca sumă de numere naturale nenule distincte (abstracţie fcând de ordinea termenilor) se foloseşte metoda backtracking obţinându-se, în ordine, toate soluţiile: **1+2+5, 1+3+4, 1+7, 2+6, 3+5**. Aplicând exact aceeaşi metodă, se determină soluţiile pentru scrierea numărului **10**. Câte soluţii de forma **1+...** există?

a. 3 **b.** 4 **c.** 5 **d.** 6

52. Se consideră mulţimile **A={1,2,3}, B={1}, C={2,3,4}**. Elementele produsului cartezian **AxBxC** se generează, folosind metoda backtracking, în ordinea **(1,1,2),(1,1,3),(1,1,4),(2,1,2),(2,1,3), (2,1,4),(3,1,2),(3,1,3),(3,1,4).** Dacă prin acelaşi algoritm se generează produsul cartezian al mulţimilor **AxBxC** unde **A={x,y}, B={x},C={x,y,z},** atunci cel de-al treilea element generat este :

a. (x,x,y) **b.** (x,y,x) **c.** (x,x,z) **d.** (x,y,z)

53. Se generează toate cuvintele obţinute prin permutarea literelor unui cuvânt dat. Astfel, pentru un cuvânt cu patru litere (nu neapărat distincte) $L_1L_2L_3L_4$, cuvintele se generează în ordinea lexicografică a permutărilor literelor: $L_1L_2L_3L_4$, $L_1L_2L_4L_3$, $L_1L_3L_2L_4$, $L_1L_3L_4L_2$, $L_1L_4L_2L_3$ etc. Dacă se generează permutările literelor cuvântului **barca** se obţin la un moment dat, în ordine, cuvintele **bacra, bacar, baarc.** Precizaţi cuvântul generat imediat înaintea acestora şi cuvântul generat imediat după ele:

a. **barac** şi **braca** **b.** **barac** şi **baacr**
c. **baacr** şi **barac** **d.** **barca** şi **baacr**

54. Generarea tuturor şirurilor de trei elemente, fiecare element putând fi oricare număr din mulţimea {**1,2,3**}, se realizează cu ajutorul unui algoritm echivalent cu algoritmul de generare a:

a. permutărilor **c.** produsului cartezian
b. combinărilor **d.** aranjamentelor

55. Utilizând metoda backtracking, se generează în ordine lexicografică, toate anagramele cuvântului **caiet.** Ştiind că primele **2** soluţii sunt **aceit** şi **aceti**, care este cuvântul generat înaintea cuvântului **tiaec** ?

a. **teica** **b.** **tieac** **c.** **ticae** **d.** **tiace**

56. Se consideră un număr natural nenul **n** având exact **k** cifre, cifrele lui fiind distincte două câte două, iar printre cele k cifre se găseşte şi cifra 0. Permutând cifrele lui **n** se obţin alte numere naturale. Câte dintre numerele obţinute, inclusiv **n**, au exact **k** cifre?

a. **k!-(k-1)!** **b.** **k!** **c.** **(k-1)!** **d.** **(k+1)!**

57. Câte numere de **10** cifre pot fi obţinute utilizând numai cifrele **0** şi **9**?

a. 2^{10} **b.** 2^9 **c.** **9** **d.** **10**

58. Utilizând metoda backtracking se generează toate posibilităţile de aranjare a **8** dame pe tabla de şah astfel încît acestea să nu se atace. Fiecare soluţie se exprimă sub forma unui vector $c=(c_1,c_2,\ldots,c_8)$ unde c_i reprezintă coloana pe care se află dama de pe linia **i.** Ştiind că primele **2** soluţii generate sunt **(1,5,8,6,3,7,2,4)**, **(1,6,8,3,7,4,2,5)** să se determine soluţia generată de algoritm imediat după soluţia **(8,2,4,1,7,5,3,6)**.

a. **(8,1,2,3,4,5,6,7)** **b.** **(8,4,2,7,6,1,3,5)**
c. **(8,2,5,3,1,7,4,6)** **d.** **(7,4,2,5,8,1,3,6)**

59. Utilizând metoda backtacking, se generează în ordine crescătoare toate numerele naturale de **5** cifre distincte, formate doar din cifrele **1,2,3,4** şi **5**. A câta soluţie generată va fi numărul **15234**?

a. **19** **b.** **18** **c.** **20** **d.** **21**

60. Se utilizează metoda Backtracking pentru a genera în ordine crescătoare, toate numerele naturale de **5** cifre distincte, care se pot forma cu cifrele **0, 1, 2, 3** şi **4.** Să se precizeze numărul generat imediat înaintea şi numărul generat imediat după secvenţa următoare : **12034, 12043, 12304, 12340**

a. **10423** şi **12403** **b.** **10423** şi **12433** **c.** **10432** şi **12403** **d.** **10432** şi **12433**

61. Dacă se utilizează metoda backtracking pentru a genera toate permutările de **4** obiecte şi primele **5** permutări generate sunt: **4 3 2 1, 4 3 1 2, 4 2 3 1, 4 2 1 3, 4 1 3 2,** atunci a **6**-a permutare este:

a. **3 4 2 1** **b.** **4 1 2 3** **c.** **3 2 1 4** **d.** **1 4 3 2**

62. Dacă se construieşte, utilizând metoda Backtracking, produsul cartezian **AxBxC** pentru mulţimile **A={1,2,3}, B={1,2}, C={1,2,3,4}**, care dintre următoarele triplete **nu** face parte din acest produs?

a. **(3,2,1)** **b.** **(1,3,2)** **c.** **(1,2,3)** **d.** **(1,1,1)**

63. Problema generării tuturor codurilor formate din **6** cifre distincte (cifre din mulţimea **{0,1,2,3,4,5,6,7,8,9}**) este similară cu generarea tuturor:

a. submultimilor cu **6** elemente ale mulţimii **{0,1,2,3,4,5,6,7,8,9}**

b. permutărilor unei mulţimi cu **6** elemente

c. aranjamentelor de **10** elemente luate câte **6**

d. elementelor produsului cartezian A^6 unde **A** este o mulţime cu **10** elemente

64. O clasă de **30** de elevi este la ora de educaţie fizică şi profesorul doreşte să formeze o echipă de **5** elevi. El îi cere unui elev să îi genereze toate posibilităţile de a forma o grupă de **5** elevi din acea clasă. Această problemă este similară cu generarea tuturor:

a. elementelor produsului cartezian A^5, **A** fiind o mulţime cu **30** de elemente

b. partiţiilor unei mulţimi

c. aranjamentelor de **30** de elemente luate câte **5**

d. combinărilor de **30** de elemente luate câte **5**

65. Într-un liceu sunt **n** clase iar în fiecare clasă sunt câte **25** de elevi. Problema determinării tuturor echipelor de **n** elevi, câte unul din fiecare clasa, este similară cu generarea tuturor:

a. elementelor produsului cartezian A^n, unde **A={1,2,…,25}**

b. submulţimilor de **n** elemente ale mulţimii **{1,2,…,25}**

c. permutărilor mulţimii **{1,2,…,n}**

d. partiţiilor mulţimii **{1,2,…,n}**

66. Se utilizează metoda backtracking pentru a determina toate modalităţile de a descompune pe **8** ca sumă de numere naturale nenule distincte (făcând abstracţie de ordinea termenilor) şi se obţin soluţiile **1+2+5**, **1+3+4**, **1+7**, **2+6**, **3+5**, **8**. Câte sume diferite, cu patru termeni, se obţin utilizând aceeaşi metodă, pentru descompunerea numărului **15**?

a. 10 **b.** 1 **c.** 6 **d.** 5

67. Se utilizează metoda backtracking pentru a determina toate modalităţile de a descompune pe **8** ca sumă de numere naturale nenule distincte (făcând abstracţie de ordinea termenilor) şi se obţin soluţiile în această ordine: **8**, **7+1**, **6+2**, **5+3**, **5+2+1**, **4+3+1**. Aplicând exact aceeaşi metodă pentru descompunerea numărului **14** în sumă de numere distincte, care este soluţia care va fi afişată imediat după soluţia **9+5**?

a. 10+3+1 **b.** 8+5+1 **c.** 9+3+2 **d.** 9+4+1

68. Se cere determinarea tuturor numerelor formate din **n** cifre distincte alese dintr-o mulţime cu **m** (**0<n≤m≤9**) cifre nenule date. Problema este echivalentă cu generarea tuturor:

a aranjamentelor de **m** obiecte luate câte **n**

b submulţimilor cu **m** elemente ale unei mulţimi cu **n** elemente

c permutărilor de **n** obiecte

d aranjamentelor de **n** obiecte luate câte **m**

69. Se consideră algoritmul care generează în ordine strict crescătoare toate numerele naturale de câte trei cifre distincte, cifrele fiind mai mici sau egale ca **4**. Precizaţi care dintre următoarele numere nu poate fi generat prin acest algoritm.

a. 123 **b.** 134 **c.** 124 **d.** 132

70. Un elev aplica metoda Backtracking pentru a genera toate submulţimile cu **k** elemente ale unei mulţimi cu **n** elemente. Dacă **n=5** şi **k=2** atunci numărul de submulţimi pe care le-a generat elevul este :

a. 60 **b.** 10 **c.** 20 **d.** 12

71. Construim anagramele unui cuvânt $L_1L_2L_3L_4$ prin generarea în ordine lexicografică a permutărilor indicilor literelor cuvântului şi obţinem $L_1L_2L_3L_4$ $L_1L_2L_4L_3$ $L_1L_3L_2L_4$ … $L_4L_3L_1L_2$ $L_4L_3L_2L_1$. Pentru anagramele cuvântului **caiet**, după şirul **caeit, caeti, catie** cuvintele imediat următoare sunt:

a. **catei** şi **ciaet** **b.** **ciaet** şi **caite**

c. **catei** şi **ciate** **d.** **ciaet** şi **ciate**

72. Folosind metoda backtracking, se generează toate numerele de **4** cifre distincte, cu proprietatea că cifrele aparţin multimii **{7,8,3,2,5}**. Primele 10 soluţii generate sunt: **7832, 7835, 7823, 7825, 7853, 7852, 7382, 7385, 7328, 7325**. Indicaţi ce număr urmează după **2538:**

a. 5783 **b.** 5782 **c.** 2537 **d.** 5738

73. Se generează în ordine crescătoare toate numerele de **4** cifre, care se pot forma cu elementele mulţimii **{0,1,2,3,4}**. Primele soluţii generate sunt, în ordine, **1000,1001,1002,1003,1004,1010,1011,1012**, … Să se precizeze numărul anterior şi cel următor secvenţei de numere consecutive: **3430,3431,3432,3433**

a. 3421 şi 3440 **c.** 3421 şi 3434
b. 3424 şi 3440 **d.** 3424 şi 3434

74. Un program generează toate cuvintele obţinute prin permutarea literelor unui cuvânt dat. Astfel, pentru un cuvânt cu 6 litere (nu neapărat distincte) $L_1L_2L_3L_4L_5L_6$, cuvintele se generează în ordinea lexicografică a permutărilor literelor: $L_1L_2L_3L_4L_5L_6$, $L_1L_2L_3L_4L_6L_5$, $L_1L_2L_3L_5L_4L_6$, $L_1L_2L_3L_5L_6L_4$, $L_1L_2L_3L_6L_4L_5$, etc. Ştiind că se aplică această metodă pentru cuvântul **examen**, care cuvânt trebuie eliminat din urmatoarea secvenţă astfel încât cele care rămân să reprezinte o succesiune corectă de cuvinte generate succesiv prin acest procedeu?

exemna, **exenam**, **exenma**, **exname**, **exnaem**, **exeman**, **exnmae**

a. exeman **b.** exenma **c.** exnaem **d.** exnmae

75. Într-un spectacol, sunt prezentate cinci melodii numerotate cu **1**, **2**, **3**, **4** şi **5**. Utilizând metoda Backtracking, se generează toate posibilităţile de a le prezenta pe toate, ştiind că melodia **1** trebuie prezentată după melodia **2** într-o ordine nu neaparat consecutivă, iar melodia **5** va fi prezentată ultima. Câte asemenea posibilităţi există?

a. 6 **b.** 30 **c.** 12 **d.** 24

76. Un algoritm Backtracking generează toate şirurile alcătuite din câte **5** cifre binare (0 şi 1). Numărul soluţiilor generate va fi egal cu:

a. 5 **b.** 32 **c.** 10 **d.** 31

77. Se generează cele 10 combinări de 5 obiecte luate câte 3: **1 2 3**, **1 2 4**, **1 2 5**, **1 3 4**, **1 3 5**, **1 4 5**, **2 3 4**, **2 3 5**, **2 4 5**, **3 4 5**. Se observă că 2 soluţii conţin în configuraţia lor secvenţa **2 4**. Pentru problema generării tuturor combinărilor de 6 obiecte luate câte 4, stabiliţi câte dintre soluţii conţin în configuraţia lor secvenţa **3 4**.

a. 2 **b.** 6 **c.** 4 **d.** 5

78. La o tombolă, la care participă **n** (**n≥4**) copii se oferă **4** premii: o minge, un arc, o carte şi o tricicletă. Ştiind că toate premiile vor fi acordate şi că niciun copil nu va primi mai mult de un premiu, ce modalităţi diferite de acordare a premiilor există?

Rezolvarea acestei probleme este echivalentă cu:

a. generarea combinărilor de **n** obiecte luate câte **4**

b. generarea aranjamentelor de **n** obiecte luate câte **4**

c. generarea permutărilor de **n** obiecte

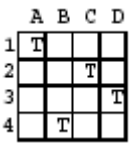
d. generarea aranjamentelor de **4** obiecte luate câte **n**

79. Se generează toate partiţiile mulţimii **{1 2 3 4 5 6}**, partiţii formate din cel puţin două submulţimi. Dintre ele, 25 au proprietatea că toate submulţimile ce formează o partiţie au acelaşi număr de elemente: **{1 2 3}{4 5 6}; {1 2 5}{3 4 6}; {1 4 5}{2 3 6}; {1 4}{2 3}{5 6}; {1 6}{2 5}{3 4}; {1}{2}{3}{4}{5}{6}**, etc.

Pentru o mulţime de 4 obiecte, câte astfel de modalităţi de partiţionare există astfel încât toate submulţimile unei partiţii să aibă acelaşi număr de elemente?

a. **3** **b.** **5** **c.** **6** **d.** **4**

80. Două ture, indiferent de culoare, se atacă dacă se află pe aceeaşi linie sau pe aceeaşi coloană. Pe o tablă cu 4 linii şi 4 coloane se aşează 4 ture, astfel încât oricare două să nu se atace între ele. O soluţie este reprezentată în figura alăturată. Ştiind că tabla nu se poate roti şi că două soluţii sunt diferite dacă diferă prin poziţia a cel puţin una din cele 4 ture stabiliţi câte soluţii distincte există.



a. **24** **b.** **16** **c.** **12** **d.** **256**

81. Se utilizează metoda backtracking pentru a genera toate cuvintele de câte două litere distincte din mulţimea **{d,a,n,s}** astfel încât să nu existe o literă **d** lângă o literă **s.** Cuvintele se obţin în ordinea: **da, dn, ad, an, as, nd, na, ns, sa, sn**. Se foloseşte aceeaşi metodă pentru a genera toate cuvintele de câte trei litere distincte din mulţimea **{d,a,n,s}** astfel încât să nu existe o literă **a** alături de o literă **s.**

Care este a patra soluţie generată?

a. **dsn** **b.** **dsa** **c.** **adn** **d.** **dns**

82. Dacă se utilizează metoda backtracking pentru a genera toate permutările mulţimii **{a,b,c,d}** şi primele soluţii afişate sunt **dcba,dcab,dbca**, atunci penultima soluţie este:

a. **acdb** **b.** **dcab** **c.** **abcd** **d.** **abdc**

83. Un şir **s** este format din **n** valori din mulţimea **{1,-1}** astfel încât suma tuturor termenilor şirului este egală cu **0** şi orice secvenţă formată din primele **p (p<n)** elemente ale şirului are proprietatea că suma componentelor secvenţei respective este un număr nenegativ.
De exemplu, pentru **n=4**, există două astfel de şiruri: **1 -1 1 -1** şi **1 1 -1 -1.**
Dacă se utilizează metoda backtracking, pentru **n=6**, numărul de şiruri **s** definite după regula de mai sus care vor fi generate este:

a. 16 **b.** 5 **c.** 8 **d.** 4

84. Având la dispoziţie cele **7** note muzicale, algoritmul de generare a tuturor succesiunilor (melodiilor) distincte formate din exact **5** note diferite este similar cu algoritmul de generare a:

a. permutărilor **b.** combinărilor **c.** produsului cartezian **d.** aranjamentelor

85. Problema generării tuturor numerelor de **n** cifre, folosind doar cifrele **1**, **5** şi **7**, este echivalentă cu problema:

a. generării produsului cartezian a **3** mulţimi cu câte **n** elemente fiecare
b. generării aranjamentelor de **n** elemente luate câte **3**
c. generării produsului cartezian a **n** mulţimi cu câte **3** elemente fiecare
d. generării combinărilor de **n** elemente luate câte **3**

86. Se generează în ordine lexicografică toate tripletele **vocală-consoană-vocală** cu litere din intervalul **A-F** al alfabetul limbii engleze: **ABA**, **ABE**, **ACA**, **ACE**, **ADA**, **ADE, AFA**, **AFE EBA**, **EBE**, **ECA**, **ECE**, **EDA**, **EDE, EFA**, **EFE**. Dacă se generează, folosind aceeaşi metodă, tripletele **consoană-vocală-consoană** cu litere din intervalul **E-P** al alfabetului limbii engleze, stabiliţi care dintre următoarele variante este o secvenţă de triplete generate unul imediat după celălalt.

a. EPA EPE EPI **b.** FON FOP GIF **c.** LOP MEF MEG **d.** PIJ PIL PIN

87. Pentru soluţionarea cărei problemele dintre cele enumerate mai jos se recomandă utilizarea metodei Backtracking ?

a. determinarea tuturor variantelor care se pot obţine din **6** aruncări consecutive cu zarul
b. determinarea reuniunii a **n** mulţimi
c. determinarea tuturor divizorilor unui număr **n**
d. determinarea tuturor elementelor mai mici decât **10000** din şirul lui Fibonacci

88. Dacă pentru generarea tuturor submulţimilor unei mulţimi **A={1,2,..n}**, cu **1≤n≤10**, se utilizează un algoritm backtracking astfel încât se afişează în ordine, pentru **n=3**, submulţimile **{},{1},{2},{3},{1,2},{1,3},{2,3},{1,2,3}**, atunci, utilizând exact acelaşi algoritm pentru **n=4**, în şirul submulţimilor generate, soluţia a **7-a** va fi:

a. {1,3} **b.** {4} **c.** {1,2,3} **d.** {1,4}

89. Se generează şiruri formate din caracterele **'A'** şi **'B'**. Dacă se utilizează un algoritm backtracking care afişează în ordine, pentru **n=3**, şirurile **BBB, BBA, BAB, BAA, ABB, ABA, AAB, AAA** atunci pentru **n=4**, după şirul **ABAA** se va afişa şirul:

a. **ABAB** **b.** **BABA** **c.** **AABA** **d.** **AABB**

90. Construim anagramele unui cuvânt $L_1L_2L_3$ prin generarea permutărilor indicilor literelor cuvântului: $L_1L_2L_3$, $L_1L_3L_2$, $L_2L_1L_3$, $L_2L_3L_1$, $L_3L_1L_2$, $L_3L_2L_1$. Pentru anagramele cuvântului **arc**, după şirul **arc,acr,rac,rca**, cuvintele imediat următoare sunt, în ordine:

a. **car,cra** **b.** **acr,car** **c.** **cra,car** **d.** **car,rac**

91. Produsul cartezian **{1,2,3}x{2,3}** este obţinut cu ajutorul unui algoritm backtracking care generează perechile **(1,2),(1,3),(2,2),(2,3),(3,2),(3,3)**.

Care este numărul perechilor obţinute prin utilizarea aceluiaşi algoritm la generarea produsului cartezian **{1,2,3,4}x{2,3,4}** ?

a. **12** **b.** **10** **c.** **81** **d.** **6**

92. Construim anagramele unui cuvânt $L_1L_2L_3$ prin generarea permutărilor indicilor literelor cuvântului: $L_1L_2L_3$, $L_1L_3L_2$, $L_2L_1L_3$, $L_2L_3L_1$, $L_3L_1L_2$, $L_3L_2L_1$. Pentru anagramele cuvântului **dac**, după şirul **dac,dca,adc,acd**, cuvintele imediat următoare sunt, în ordine:

a. **cda,dca** **b.** **cad,cda** **c.** **adc,cad** **d.** **cda,cad**

93. Un elev realizează un program care citeşte o valoare naturală pentru o variabilă **n** şi apoi generează şi afişează toate permutările mulţimii **1,2,...,n**. Rulând programul pentru **n=3**, permutările apar în următoarea ordine: **3 2 1, 3 1 2, 2 3 1, 2 1 3, 1 3 2, 1 2 3**. Dacă va rula din nou programul şi va introduce pentru variabila **n** valoarea **5**, **imediat după** permutarea **4 1 2 3 5**, programul va afişa permutarea

a. **3 5 4 2 1** **b.** **4 5 3 2 1** **c.** **4 1 2 5 3** **d.** **3 5 4 3 2**

94. Considerăm **n** copii şi **p** tricouri pe care sunt imprimate numerele de la **1** la **p** (**n,p∈N**, **1≤p≤n**). Algoritmul care să genereze şi să afişeze toate modurile în care pot fi împărţite cele **p** tricouri celor **n** copii este echivalent cu algoritmul folosit pentru generarea:

a. **aranjamentelor** **c.** produsului cartezian
b. permutărilor **d.** combinărilor

95. Câte grupuri formate din câte **4** elevi se pot realiza din cei **n** elevi ai unei clase (**n≥4**)?

a. P_4 **b.** A_4^n **c.** C_4^n **d.** C_n^4

96. Un program citeşte un număr natural nenul, generează toate modurile distincte în care numărul dat poate fi scris ca sumă de cel puţin două numere naturale nenule distincte şi afişează numărul soluţiilor obţinute. Două sume se consideră distincte dacă diferă prin cel puţin un termen. De exemplu, pentru numărul **8** vor fi generate sumele **1+2+5**, **1+3+4**, **1+7**, **2+6** şi **3+5**, deci se va afişa **5**. Care este valoarea afişată de către program dacă numărul citit este **10**?

a. **20** **b.** **42** **c.** **10** **d.** **9**

97. Un program generează toate cuvintele obţinute prin permutarea literelor unui cuvânt dat. Astfel, pentru un cuvânt cu 4 litere (nu neapărat distincte) $L_1L_2L_3L_4$, cuvintele se generează în ordinea lexicografică a permutărilor literelor: $L_1L_2L_3L_4$, $L_1L_2L_4L_3$, $L_1L_3L_2L_4$, $L_1L_3L_4L_2$, $L_1L_4L_2L_3$, etc. Pentru cuvântul **"mama"**, **imediat după** prima apariţie a cuvântului **"mmaa"**programul va afişa cuvântul:

a. **mama** **b.** **mmaa** **c.** **maam** **d.** **maam**

5.2. Probleme

1. Se citesc două numere naturale: **n** (**1≤n≤20**) şi **k** (**1≤k≤9**). Să se scrie un program care să afişeze câte numere naturale care îndeplinesc următoarele cerinţe există:

- au cel mult **n** cifre;
- sunt formate numai din cifrele **1** şi **0**;
- încep obligatoriu cu cifra **1**;
- conţin exact **k** cifre de **1**.

Exemplu: pentru **n = 4** şi **k = 3**, programul va afişa valoarea **4** deoarece sunt patru numere care îndeplinesc cerinţele impuse; acestea sunt **111**, **1011**, **1101**, **1110**. Alegeţi o metodă eficientă de rezolvare din punct de vedere al timpului de executare.

2. Fie **M = {1,2,3,4,5,6,7,8,9,10}** mulţimea formată din primele **10** numere naturale nenule. Scrieţi un program Pascal eficient din punct de vedere al timpului de rulare şi al spaţiului de memorie utilizat, care citeşte de la tastatură o valoarea naturală **k**, (**1≤k≤6**) şi apoi afişează **12** permutări ale mulţimii **M** care îndeplinesc proprietatea că numerele **k,k+1,...,k+4** apar în fiecare dintre aceste **12** permutări în poziţii consecutive şi în această ordine. De exemplu, pentru **k = 3**, una dintre permutările care îndeplineşte această proprietate este permutarea

1 9 2 10 3 4 5 6 7 8

Fiecare permutare va fi afişată pe câte o linie a ecranului.

6. Structuri dinamice de date

6.1. Teste grilă (Pascal | C / C++)

1. Variabila **p** este declarată astfel:

`var p:^integer;`	`int *p;`

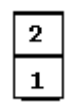
Ştiind că **p** reţine adresa unei informaţii de tip **integer/int**, care dintre următoarele instrucţiuni afişează valoarea respectivei informaţii întregi?

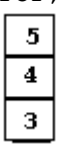
a.	`writeln(p)`	**a.**	`cout << p; / printf("%p",p);`
b.	`writeln(^p)`	**b.**	`cout << p*; / printf("%d",p*);`
c.	`writeln(p^)`	**c.**	`cout << *p; / printf("%d",*p);`
d.	`writeln(@p)`	**d.**	`cout << &p; / printf("%d",&p);`

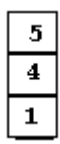
2. Se consideră o stivă **S** iniţial vidă. În stivă se introduc în această ordine elementele: **3, 5, 6, 7, 10, 13**. Se fac apoi următoarele operaţii: se scoate un element din stivă, apoi se adaugă elementul cu valoarea **8**, se scot apoi două elemente din stivă. Ce element se va găsi în vârful stivei?

a. 8 **b.** 7 **c.** 6 **d.** 10

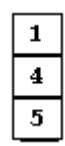
3. Se consideră o stivă în care iniţial au fost introduse, în această ordine, valorile **1** şi **2**, ca în figura alăturată. Dacă se notează cu **PUSH(x)** operaţia prin care se adaugă valoarea **x** în vârful stivei, şi **POP** operaţia prin care se extrage elementul din vârful stivei, care este conţinutul acesteia în urma operaţiilor
POP;PUSH(3);POP;PUSH(4);PUSH(5)?



a.


b.


c.

d.


4. Se consideră o stivă alocată dinamic care are cel puţin **10** elemente. Variabila **vf** memorează adresa de început a stivei şi orice element al stivei memorează în câmpul **info** un număr întreg, iar în câmpul **next** adresa nodului următor

```
while
(vf^.info mod2=0)and(vf<>nil)do
begin
 aux:=vf;
 vf:=aux^.next;dispose(aux)
end;
```

```
while
(vf && vf->info%2==0)
{
 aux=vf;
 vf=aux->next;
 delete aux;  /free(aux);
}
```

Dacă în urma executării secvenţei alăturate de program, variabila **vf** are valoarea **NULL**, atunci:

a. primul element memorat în stivă este par, celelalte fiind numere impare
b. în stivă nu s-a memorat nici un număr impar
c. ultimul element memorat în stivă este par celelalte fiind numere impare
d. în stivă nu s-a memorat nici un număr par

5. Se consideră o stivă alocată dinamic. Tipul elementelor care alcătuiesc stiva este cel din definiţia următoare.

```
type lista=^element;
      element=record
           nr:integer; adr:lista
      end;
var vf,p: lista;
```

```
struct element
 {
    int nr;
    element *adr;
 } *vf, *p;
```

Adresa elementului din vârful stivei este reţinută de variabila **vf.** Dacă în stivă se află cel puţin un element, care dintre următoarele secvenţe de instrucţiuni determină, în urma executării, eliminarea unui element din stivă?

a.
```
vf:=vf^.adr; dispose (vf);
```

b.
```
p:=vf^.adr ; vf:=vf^.adr;
dispose (p);
```

c.
```
p:=vf; vf:=p^.adr;
dispose (p);
```

d.
```
p:=vf; p:=vf^.adr;
dispose (p);
```

a.
```
vf=vf->adr;
delete vf; / free(vf);
```

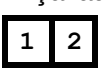
b.
```
p=vf->adr; vf=vf->adr;
delete p; / free(p);
```

c.
```
p=vf; vf=p->adr;
delete p; / free(p);
```

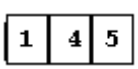
d.
```
p=vf; p=vf->adr;
delete p; / free(p);
```

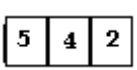
6. Se consideră o coadă **C** iniţial vidă. În coadă se introduc în această ordine elementele: **3, 5, 6, 7, 10, 13**. Se fac apoi următoarele operaţii: se elimină un element din coadă, apoi se adaugă elementul cu valoarea **8**, se elimină apoi două elemente din coadă. Care va fi ultimul element al cozii?

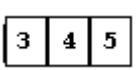
a. 3 **b.** 7 **c.** 13 **d.** 8

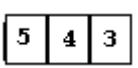
7. Se consideră o coadă în care iniţial au fost introduse, în această ordine, elementele **1** şi **2**: 

Dacă se notează cu **AD(x)** operaţia prin care se adaugă informaţia **x** în coadă, şi cu **EL** operaţia prin care se elimină un element din coadă, care este rezultatul executării secvenţei **EL;AD(3);EL;AD(4);AD(5);** ?

a. 

b. 

c. 

d. 

8. Se consideră declarările următoare utilizate pentru a defini o listă simplu înlănţuită:

```
type lista=^nod;
     nod=record
           nr:integer; adr:lista
         end;
var prim,p: lista;
```

```
struct nod
     { int nr;
       nod* adr;} *prim, *p;
```

Variabila **prim** memorează adresa primului element al listei. Afişarea informaţiei din cel de-al treilea element al listei se realizează prin executarea instrucţiunii:

a.
```
p:=prim^.adr;
write(p^.adr^.adr^.nr);
```

b.
```
write(prim^.nr);
```

c.
```
write(prim^.adr^.adr^.nr);
```

d.
```
write(prim^.adr^.nr);
```

a.
```
p=prim->adr;
cout<<p->adr->adr->nr;
/ printf("%d", p->adr->adr->nr);
```

b.
```
cout<<prim->nr;
/ printf("%d", prim->nr);
```

c.
```
cout<<prim->adr->adr->nr;
/printf("%d",prim->adr->adr-
>nr);
```

d.
```
cout<<prim->adr->nr;
/ printf("%d", prim->adr->nr);
```

9. Care dintre următoarele afirmaţii este corectă pentru o listă simplu înlănţuită alocată dinamic?

a. Numărul de elemente din listă nu poate fi modificat pe parcursul unui program.

b. Numărul de elemente din listă poate fi modificat pe parcursul unui program.

c. Nu este permisă eliminarea primului element din listă.

d. Nu este permisă inserarea unui element înaintea primului element din listă.

10. O listă simplu înlănţuită conţine 4 componente, fiecare nod al listei reţinând în câmpul **next** adresa nodului următor. Dacă variabila **p**, ce memorează adresa unui nod din listă, verifică relaţia:

```
p^.next^.next=nil
```

```
p->next->next==NULL
```

atunci nodul ce precede nodul de la adresa **p** în listă este:

a. al doilea **b.** ultimul **c.** penultimul **d.** primul

11. Într-o listă simplu înlănţuită alocată dinamic fiecare element reţine în câmpul **nr** un număr întreg şi în câmpul **urm** adresa următorului element din listă. Ştiind că variabila **p** conţine adresa primului element din listă şi variabila **t** este de acelaşi tip cu variabila **p**, stabiliţi care dintre următoarele secvenţe eliberează întreaga zonă de memorie ocupată de elementele listei:

a.
```
while (p<>nil) do begin
   t:=p; p:=p^.urm;
   dispose(p)
end
```
b.
```
while (p<>nil) do begin
   t:=p; p:=p^.urm;
   dispose(t)
end
```
c.
```
while (p<>nil) do begin
   t:=p; t:=t^.urm;
   dispose(t)
end
```
d.
```
dispose(p)
```

a.
```
while(p!=NULL)
{ t=p; p=p->urm;
  delete p;  / free(p);
}
```
b.
```
while(p!=NULL)
{  t=p; p=p->urm;
   delete t;  / free(t);
}
```
c.
```
while(p!=NULL)
{  t=p; t=t->urm;
   delete t;  / free(t);
}
```
d.
```
delete p; / free(p);
```

12. Fiecare element al unei liste liniare simplu înlănţuită alocată dinamic reţine în câmpul **adru** adresa elementului următor din listă. Dacă **p** reţine adresa primului element, iar lista are cel puţin două elemente, care dintre următoarele secvenţe de instrucţiuni şterge al doilea element al listei?

a.
```
q:=p^.adru;p^.adru:=q^.adru;
dispose(q)
```
b.
```
p^.adru:=p^.adru^.adru;
dispose(p^.adru)
```
c.
```
q:=p^.adru; dispose(q);
p^.adru:=q^.adru
```
d.
```
dispose(p^.adru)
```

a.
```
q=p->adru;p->adru=q->adru;
delete q;
```
b.
```
p->adru=p->adru->adru;
delete p->adru;
```
c.
```
q=p->adru; delete q;
p->adru=q->adru;
```
d.
```
delete p->adru;
```

13. Intr-o listă liniară simplu înlanţuită alocată dinamic fiecare element conţine în câmpul **nume** numele şi prenumele unei persoane, iar în câmpul **urm** adresa elementului următor. Ştiind că variabila **x** memorează adresa unui element din listă, cum procedăm pentru a verifica dacă valoarea câmpului **nume** al variabilei **x** este egală cu şirul de caractere **"Popescu Ion"**?

Varianta Pascal	**a.** `if pos(x^.nume,'Popescu Ion')=1 then`
	b. `if x='Popescu Ion' then`
	c. `if x^.nume='Popescu Ion' then`
	d. `if x.nume='Popescu Ion' then`
Varianta C/C++	**a.** `if(x->nume=="Popescu Ion")`
	b. `if(strcmp(x,"Popescu Ion")==0)`
	c. `if(strcmp(x->nume,"Popescu Ion")==0)`
	d. `if(x.nume=="Popescu Ion")`

14. Într-o listă simplu înlănţuită, cu cel puţin patru elemente, fiecare element reţine în câmpul **urm** adresa elementului următor din listă. Dacă **p**, **q** şi **r** sunt adresele a trei elemente din listă astfel încât:

`p^.urm=q^.urm^.urm`	`p->urm==q->urm->urm`
`r^.urm=q`	`r->urm==q`

atunci ordinea logică a elementelor în listă (elementele fiind identificate prin adrese) este:

a. `q, r, p` **b.** `p, r, q` **c.** `r, q, p` **d.** `p, q, r`

15. Într-o listă simplu înlănţuită, cu cel puţin patru elemente, fiecare element reţine în câmpul **urm** adresa elementului următor din listă. Dacă **p**, **q** şi **r** sunt adresele a trei elemente din listă astfel încât:

`q=p^.urm^.urm`	`q==p->urm->urm`
`r^.urm=p^.urm^.urm`	`r->urm==p->urm->urm`

atunci ordinea logică a elementelor în listă (elementele fiind identificate prin adrese) este:

a. `q, r, p` **b.** `r, q, p` **c.** `p, r, q` **d.** `p, q, r`

16. Într-o listă simplu înlănţuită, cu cel puţin patru elemente, fiecare element reţine în câmpul **adr** adresa elementului următor din listă, iar **q** este adresa ultimului element din listă. Atunci **p** este adresa antepenultimului element din listă dacă şi numai dacă este satisfăcută condiţia:

a. `q^.adr^.adr=p`	**a.** `q->adr->adr==p`
b. `p^.adr=q`	**b.** `p->adr==q`
c. `p^.adr^.adr=q`	**c.** `p->adr->adr==q`
d. `q^.adr=p^.adr^.adr`	**d.** `q->adr==p->adr->adr`

17. Într-o listă simplu înlănţuită (cu cel puţin patru elemente) fiecare element reţine în câmpul **urm** adresa elementului următor din listă, iar **p** memorează adresa celui de-al treilea element din listă. Atunci **q** reţine adresa primului element din listă dacă şi numai dacă este satisfăcută condiţia:

a. `p^.urm^.urm=q^.urm`	**a.** `p->urm->urm==q->urm`
b. `p^.urm^.urm=q`	**b.** `p->urm->urm==q`
c. `q^.urm^.urm^.urm=p^.urm`	**c.** `q->urm->urm->urm==p->urm`
d. `q^.urm^.urm=p^.urm`	**d.** `q->urm->urm==p->urm`

18. Într-o listă simplu înlănţuită, cu cel puţin două elemente, fiecare element reţine în câmpul **urm** adresa elementului următor din listă, iar **q** memorează adresa penultimului element din listă. Dacă **p** reţine adresa unui element ce urmează a fi adăugat la sfârşitul listei şi **p^.urm** / **p->urm** are valoarea **nil/ NULL**, stabiliţi care dintre următoarele este o operaţie corectă de adăugare:

a. `p^.urm:=q`	**a.** `p->urm=q;`
b. `q^.urm:=p`	**b.** `q->urm=p;`
c. `q^.urm^.urm:=p`	**c.** `q->urm->urm=p;`
d. `p^.urm^.urm:=q`	**d.** `p->urm->urm=q;`

19. Se consideră că variabila **prim** memorează adresa de început a unei liste liniare simplu înlănţuite nevide. Orice element al listei memorează în câmpul **urm** adresa elementului următor din listă. Dacă expresia **prim^.urm | prim->urm** este diferită de **nil|NULL** şi expresia **prim ^.urm ^.urm | prim ->urm ->urm** are valoarea **nil |NULL** atunci numărul de elemente din listă este egal cu :

a. 1 **b.** 0 **c.** 3 **d.** 2

20. Se consideră o listă simplu înlănţuită în care fiecare element memorează în câmpul **nr** un număr natural strict pozitiv şi în câmpul **urm** adresa elementului următor din listă. Lista memorează, în ordine, pornind de la primul element, valorile **1,2,2,3,3,3,4,4,4,4,5,5,5,5,5.** Ştiind că **prim** reţine adresa primului element al listei iar **p** este o variabilă de acelaşi tip cu **prim**, stabiliţi câte valori nule există în listă după executarea secvenţei:

```
p:=prim;
while p^.urm <> nil do
if p^.nr=p^.urm^.nr then
   begin
    p^.nr:=0;p:=p^.urm^.urm
   end
else p:=p^.urm;
```

```
p=prim;
while(p->urm!=NULL)
 if(p->nr == p->urm->nr)
   {p->nr=0; p=p->urm->urm;}
 else p=p->urm;
```

a. 11 **b.** 5 **c.** 6 **d.** 7

21. Într-o listă simplu înlănţuită nevidă, fiecare element reţine în câmpul **urm** adresa elementului următor din listă. Variabilele **prim** şi **p**, sunt de tip referinţă/pointer astfel declarate încât pot reţine adresa oricărui element al listei. Ştiind că variabila **prim** reţine adresa primului element din listă, cu ce trebuie înlocuite cele trei puncte din secvenţa alăturată astfel încât ea sa calculeze corect în variabila **nr** declarată de tip **longint/long**, numărul de elemente din această listă.

```
p := prim;
nr := 0;
while ...
Begin
  nr := nr + 1;p := p^.urm
End
```

```
p = prim;
nr = 0;
while (...)
{ nr = nr + 1; p = p->urm;};
```

a. `p^.urm <> nil`	a. `p->urm != NULL`
b. `prim^.urm <> nil`	b. `prim->urm != NULL`
c. `prim <> nil`	c. `prim != NULL`
d. `p <> nil`	d. `p != NULL`

22. Într-o listă simplu înlănţuită cu cel puţin trei elemente, fiecare element reţine în câmpul **`nr`** un număr întreg şi în câmpul **`urm`** adresa următorului element din listă. Dacă variabila **`prim`** reţine adresa primului element din listă, stabiliţi care dintre următoarele secvenţe afişează suma tuturor numerelor memorate în listă, mai puţin cele reţinute de primul şi ultimul element:

a.
```
s:=0;p:=prim;
while(p^.urm<>nil) do
begin p:=p^.urm;
      s:=s+p^.nr end;
write(s)
```

b.
```
s:=0;p:=prim;
while(p<>nil) do begin
s:=s+p^.nr; p:=p^.urm
end;
write(s)
```

c.
```
s:=0;p:=prim^.urm;
while(p<>nil) do begin
s:=s+p^.nr; p:=p^.urm
end;
write(s)
```

d.
```
s:=0;p:=prim;
while(p^.urm<>nil) do
begin p:=p^.urm;
      s:=s+p^.nr end;
write(s-p^.nr)
```

a.
```
s=0;p=prim;
while(p->urm!=NULL)
   {p=p->urm; s=s+p->nr;}
cout<<s;  / printf("%d",s);
```

b.
```
s=0;p=prim;
while(p!=NULL)
   {s=s+p->nr; p=p->urm;}
cout<<s;  / printf("%d",s);
```

c.
```
s=0;p=prim->urm;
while(p!=NULL)
    {s=s+p->nr; p=p->urm;}
cout<<s;  / printf("%d",s);
```

d.
```
s=0;p=prim;
while(p->urm!=NULL)
    {p=p->urm; s=s+p->nr;}
cout<<s-p->nr;
/ printf("%d",s-p->nr);
```

23. Fie o listă liniară simplu înlănţuită în care fiecare element reţine în câmpul **`urm`** adresa elementului următor din listă, iar în câmpul **`info`** informaţia utilă, de tip **`integer/int`**. Variabila **`p`** reţine adresa primului element al listei, **`p^.info`** (Pascal), respectiv **`p->info`** (C/C++), are valoarea **`5`**, ultimul element al listei reţine în câmpul **`info`** valoarea **`10`** şi lista are forma: **`5 → 2 → 4 → 7 → 8 → 10`**. Ce se afişează în urma executării secvenţei de instrucţiuni?

```
while p^.urm<>nil do
begin
 if p^.info mod 2=0 then
  write(p^.info, ' ');
  p:=p^.urm
 end;
```

```
while(p->urm!=NULL)
{ if (p->info%2==0)
   cout<<p->info<<" ";
    / printf("%d ",p->info);
  p=p->urm;
}
```

a. `2 4 8 10` **b.** `8 4 2` **c.** `10 8 4 2` **d.** `2 4 8`

24. Într-o listă simplu înlănţuită alocată dinamic de cel puţin 3 elemente, fiecare element reţine în câmpul **`adr`** adresa elementului următor. Dacă variabilele **`p`**,**`q`** şi **`u`** reţin adresele unor elemente din această listă astfel încât:

`u^.adr^.adr=q` şi `p^.adr=q`	`u->adr->adr==q` şi `p->adr==q`

atunci este adevărată relaţia:

a. `p=q`	**a.** `p==q`
b. `u^.adr<>p`	**b.** `u->adr!=p`
c. `q^.adr=p`	**c.** `q->adr==p`
d. `u^.adr=p`	**d.** `u->adr==p`

25. Într-o listă liniară simplu înlănţuită fiecare element reţine în câmpul **`adr`** adresa elementului următor din listă, iar în câmpul **`inf`** un număr întreg. Inserarea în listă a unui nou element indicat de pointerul **`p`** după un element din listă indicat de pointerul **`q`** se realizează cu secvenţa de instrucţiuni:

a. `p^.adr:=q; q^.adr:=p`	**a.** `p->adr=q; q->adr=p;`
b. `q^.adr:=p;p^.adr:=q^.adr`	**b.** `q->adr=p; p->adr=q->adr;`
c. `p^.adr:=q^.adr;q^.adr:=p`	**c.** `p->adr=q->adr;q->adr=p;`
d. `p:=q; p^.adr=q^.adr`	**d.** `p=q; p->adr=q->adr;`

26. Într-o listă simplu înlănţuită, cu cel puţin patru elemente, fiecare element reţine în câmpul **`adr`** adresa elementului următor din listă iar în câmpul **`nr`** un număr întreg. Ştiind că **`p`** reţine adresa unui element din interiorul listei, iar variabila **`q`** este de acelaşi tip cu variabila **`p`**, pentru ştergerea elementului următor celui de la adresa **`p`**, se va folosi secvenţa:

a. `q:=p^.adr; dispose(q);` `p^.adr:=p^.adr^.adr;`	**a.** `q=p->adr; delete q; / free(q);` `p->adr=p->adr->adr;`

b. `q:=p^.adr;`
`p^.adr=p^.adr^.adr;`
`dispose(q);`
	b. `q=p->adr; p->adr=p->adr->adr;`
	`delete q; / free(q);`

c. `dispose(p^.adr);`	**c.** `delete p->adr; / free(p->adr);`
d. `p^.adr:=p^.adr^.adr;` `dispose(p);`	**d.** `p->adr=p->adr->adr;` `delete p; / free(p);`

27. Într-o listă liniară simplu înlănţuită cu cel puţin **5** noduri, fiecare element reţine în câmpul **`urm`**, adresa următorului element din listă. Dacă **`p`** reţine adresa primului element din listă, ca urmare a executării căreia dintre secvenţele de mai jos, **`p`** va reţine adresa elementului al cincilea din listă?

a.
`i:=1;`
`while i<5 do begin`
`p:=p^.urm; i:=i+1`
`end`
	a.
	`i=1;`
	`while (i<5)`
	`{ p=p->urm; i++; }`

b.
`i:=1;`
`repeat p:=p^.urm; i:=i+1 until i>5`
	b.
	`i=1;`
	`do{ p=p->urm; i++;`
	`}while (i<=5);`

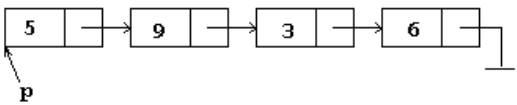
c.
```
for i:=1 to 5 do p:=p^.urm
```

d.
```
i:=1;
repeat p:=p^.urm; i:=i+1 until
i<=5
```

c.
```
for (i=1;i<=5;i++)
  p=p->urm;
```

d.
```
i=1;
do{ p=p->urm; i++; }
while (i>5);
```

28. Într-o listă liniară simplu înlănţuită, fiecare element reţine în câmpul **urm** adresa următorului nod din listă, iar în câmpul **inf** un număr întreg. Adresa primului element al listei este reţinută în variabila **p**. Dacă în listă sunt memorate, în această ordine, numerele **5 9 3 6** ca în figura de mai jos



ce numere vor fi memorate în ordine în listă în urma executării următoarei secvenţe de instrucţiuni

```
q:=p^.urm^.urm;
p^.urm^.urm:=q^.urm;
q^.urm:=p^.urm;
p^.urm:=q
```

```
q=p->urm->urm;
p->urm->urm=q->urm;
q->urm=p->urm;
p->urm=q;
```

a. 9,5,3,6 **b.** 5,9,6,3 **c.** 5,3,6,9 **d.** 5,3,9,6

29. O listă liniară simplu înlănţuită formată dintr-un număr <u>impar</u> de cel puţin **5** noduri are adresa primului nod memorată în variabila **prim**. În câmpul **ref** al fiecărui nod al listei se memorează adresa următorului nod din listă. Adresa cărui nod va fi memorată în variabila **p**, după executarea secvenţei de program de mai jos?

```
p:=prim;
q:=prim;
while q^.ref<>NIL do
begin
  q:=q^.ref^.ref; p:=p^.ref
end
```

```
p=prim; q=prim;
while (q->ref!=0)
{ q=q->ref->ref; p=p->ref;
}
```

a. Penultimul nod al listei. **b.** Nodul aflat în mijlocul listei.

c. Ultimul nod al listei. **d.** Nodul al treilea din listă.

30. Într-o listă liniară simplu înlănţuită, alocată dinamic, fiecare element reţine în câmpul **next** adresa următorului nod din listă, iar în câmpul **info** un număr întreg. Adresa primului element al listei este memorată în variabila **prim.** Se ştie că lista are cel puţin **3** noduri. Care dintre următoarele secvenţe de instrucţiuni elimină corect penultimul element al listei?

Pascal	**a.**
	`p:=prim;`
	`repeat p:=p^.next until p^.next^.next^.next=NIL;`
	`p^.next:=p^.next^.next;`
	b.
	`p:=prim; while p^.next^.next^.next<>NIL do p:=p^.next;`
	`p^.next:=p^.next^.next;`
	c.
	`p:=prim; while p^.next^.next<>NIL do p:=p^.next;`
	`p^.next:=p^.next^.next;`
	d.
	`prim^.next:=prim^.next^.next;`

C/C++	**a.**
	`p=prim; do p=p->next;`
	`while(p->next->next->next!=NULL);`
	`p->next=p->next->next;`
	b.
	`p=prim; while (p->next->next->next!=NULL) p=p->next;`
	`p->next=p->next->next;`
	c.
	`p=prim; while (p->next->next!=NULL) p=p->next;`
	`p->next=p->next->next;`
	d.
	`prim->next=prim->next->next;`

31. Într-o listă liniară simplu înlănţuită, alocată dinamic, fiecare element reţine în câmpul **next** adresa următorului nod din listă, iar în câmpul **info** un număr întreg. Adresa primului element al listei este memorată în variabila **prim**. Lista conţine cel puţin **3** noduri. Care este efectul executării următoarei secvenţe de program?

```
p:=prim; q:=p^.next^.next;
while q^.next<>NIL do begin
  p:=p^.next; q:=q^.next end;
p^.next:=q;
```

```
p=prim; q=p->next->next;
while (q->next!=NULL) {
  p=p->next; q=q->next; }
p->next:=q;
```

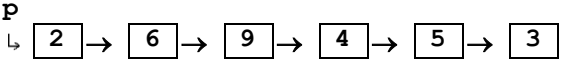
a. Eliminarea nodului din mijlocul listei.

c. Eliminarea din listă a penultimului element.

b. Eliminarea din listă a ultimului element.

d. Eliminarea celui de al doilea nod al listei

32. Într-o listă simplu înlănţuită fiecare element reţine în câmpul **adr** adresa elementului următor din listă, iar în câmpul **inf** un număr întreg. Adresa primului element al listei este memorat în variabila **p**. Dacă în listă sunt memorate, în această ordine, numerele



Ştiind că variabila **c** este de acelaşi tip cu **inf**, în urma executării secvenţei de instrucţiuni:

```
c:=p^.adr^.inf;
p^.adr^.inf:=p^.adr^.adr^.inf;
p^.adr^.adr^.inf:=c
```

```
c=p->adr->inf;
p->adr->inf=p->adr->adr->inf;
p->adr->adr->inf=c;
```

în listă vor fi memorate în ordine numerele:

a. 2 6 9 4 5 3 **b.** 6 9 4 5 3 2

c. 2 9 6 4 5 3 **d.** 6 2 9 4 5 3

33. Se consideră lista simplu înlănţuită în care fiecare nod memorează în câmpul **nr** o valoare întreagă şi în câmpul **urm** adresa nodului următor. O variabilă ce reţine adresa unui nod este de tipul **adnod**.
În listă sunt memorate, în această ordine, valorile **1,2,3,4,5**. Dacă variabila **p** reţine adresa primului nod din listă, ce se va afişa în urma apelului **f(p)**?

```
procedure f(p:adnod);
begin
 if p<>NIL then  begin
      f(p^.urm);
      write(p^.nr)
 end
end;
```

```
void f(nod*p)
{if(p!=NULL)
 {f(p->urm);
   printf("%d",p->nr);
     /  cout<<p->nr;
  }
}
```

a. 12345 **b.** 54321 **c.** 5 **d.** 1

34. O listă simplu înlănţuită nevidă alocată dinamic conţine cel puţin 3 elemente, fiecare element reţinând în câmpul **urm** adresa elementului următor din listă. Ştiind că **p** este adresa primului element şi că

```
p^.urm^.urm^.urm<>nil,
```

```
p->urm->urm->urm !=NULL,
```

deduceţi care este numărul de componente ale listei.

a. 2 **b.** cel puţin 4 **c.** 1 **d.** 3

35. Se consideră lista simplu înlănţuită în care fiecare nod memorează în câmpul **nr** o valoare întreagă şi în câmpul **urm** adresa nodului următor. Un nod al listei este de tipul **nod**.
În listă sunt memorate, în această ordine, valorile **1,2,3,4,5,6,7**. Dacă variabila **p** reţine adresa primului element din listă, ce se va returna la apelul **f(p)**?

```
function f(p:adnod):integer;
begin
 if p=nil then f:=0
 else
   if p^.nr mod 2=0 then
        f:=1+f(p^.urm)
   else f:=f(p^.urm)
end;
```

```
int f(nod *p)
{if(p==NULL) return 0;
 else
  if(p->nr%2==0)
    return 1+f(p->urm);
  else
    return f(p->urm);
}
```

a. 7 **b.** 3 **c.** 12 **d.** 0

36. Într-o listă liniară simplu înlănţuită fiecare element reţine în câmpul **urm** adresa următorului nod din listă, iar în câmpul **info** un număr întreg. Adresa primului nod al listei este memorată în variabila **p**. Dacă în listă sunt memorate în această ordine numerele **7, 8, 9, 2, 0, 2, 9, 8, 7,** ce se va afişa în urma executării secvenţei de program alăturate?

```
nr:=0;
while (p^.urm^.info<>0)
       and (p <> Nil) do
  begin
   p := p^.urm; nr:=nr+1
  end;
write(nr);
```

```
int nr = 0;
while(p->urm->info!=0&&p)
  {
      p = p->urm;
      nr++;
  }
printf("%d",nr);|cout<<nr;
```

a. 5 **b.** 2 **c.** 4 **d.** 3

37. Într-o listă liniară simplu înlănţuită cu cel puţin 3 noduri, fiecare element reţine în câmpul **urm**, adresa următorului element din listă. Dacă în variabila **p** se reţine adresa primului element din listă şi **q** este o variabilă de acelaşi tip cu **p**, atunci care din secvenţele de mai jos elimină din listă al doilea nod ?

a.
```
q:=p^.urm^.urm;
p^.urm:=q^.urm;
dispose q;
```
b.
```
q:=p^.urm;
dispose(q);
p^.urm:=q^.urm;
```
c.
```
dispose(p^.urm);
p^.urm:=p^.urm^.urm;
```
d.
```
q:=p^.urm;
p^.urm:=q^.urm;
dispose(q);
```

a.
```
q=p->urm->urm;
p->urm=q->urm;
delete q;
```
b.
```
q=p->urm;
delete q;
p->urm=q->urm;
```
c.
```
delete p->urm;
p->urm=p->urm->urm;
```
d.
```
q=p->urm;
p->urm=q->urm;
delete q;
```

38. Se consideră două liste simplu înlănţuite, nevide, alocate dinamic. Fiecare element al acestor liste reţine în câmpul **adr** adresa elementului următor în listă. Variabila **p1** reţine adresa primului element din prima listă, iar variabila **p2** reţine adresa primului element din a doua listă. Care dintre variantele următoare realizează concatenarea celor două liste?

a.
```
q:=p1;
while q<>nil do q:=q^.adr;
p2^.adr:=q;
```
b.
```
q:=p1;
while q<>nil do q:=q^.adr;
q^.adr:=p2;
```
c.
```
q:=p1;
while q^.adr<>nil do q:=q^.adr;
q^.adr:=p2;
```
d.
```
q:=p1;
while q^.adr<>nil do q:=q^.adr;
p2^.adr:=q^.adr;
```

a.
```
q=p1;
while(q!=NULL)q=q->adr;
p2->adr=q;
```
b.
```
q=p1;
while(q!=NULL) q=q->adr;
q->adr=p2;
```
c.
```
q=p1;
while(q->adr!=NULL)
q=q->adr;
q->adr=p2;
```
d.
```
q=p1;
while(q->adr!=NULL)
 q=q->adr;
p2->adr=q->adr;
```

39. Se consideră o listă liniară simplu înlănţuită ale cărei noduri reţin în câmpul **urm** adresa nodului următor sau **NULL/NIL** dacă nu există un element următor. Lista are cel puţin două elemente. Ştiind că **p** reţine adresa primului nod din listă iar **u** adresa ultimului nod, care este atribuirea corectă prin care lista liniară se transformă intr-o listă circulară?

a.	`u:=p;`	**b.**	`u^.urm:=p;`	**a.**	`u=p;`	**b.**	`u->urm=p;`
c.	`p^.urm:=u;`	**d.**	`p:=u;`	**c.**	`p->urm=u;`	**d.**	`p=u;`

40. Se consideră o listă liniară simplu înlănţuită ale cărei noduri reţin în câmpul **urm** adresa nodului următor sau **NIL/NULL** dacă nu există un element următor în listă. Lista are cel puţin un element. Ştiind că variabila **p** reţine adresa primului nod din listă, care este secvenţa prin care se inserează după nodul indicat de **p** un nou nod indicat de variabila **q**?

a.	`p:=q^.urm; q^.urm:=p^.urm;`	**a.**	`p=q->urm; q->urm=p->urm;`
b.	`p^.urm:=q; q^.urm:=p^.urm;`	**b.**	`p->urm=q; q->urm=p->urm;`
c.	`q^.urm:=p^.urm; p^.urm:=q;`	**c.**	`q->urm=p->urm; p->urm=q;`
d.	`p:=q; q^.urm:=p^.urm;`	**d.**	`p=q; q->urm=p->urm;`

41. Se consideră o listă liniară simplu înlănţuită ale cărei noduri reţin în câmpul **info** o valoare numerică întreagă. Ştiind că **r, s** şi **t** reţin adresele unor elemente din listă, stabiliţi ce se va afişa după executarea următoarei secvenţe de program:

```
r^.info:=1;
t:=r;
s^.info:=4;
s:=t;
write(r^.info,s^.info,t^.info);
```

```
r->info=1; t=r;
s->info=4; s=t;
cout<<r->info<<s->info;
    / printf("%d%d",r->info,s->info);
cout<<<<t->info;
    / printf("%d",t->info);
```

a. `144` **b.** `141` **c.** `111` **d.** `441`

42. Fie o listă liniară simplu înlănţuită ale cărei noduri reţin in câmpul **urm** adresa nodului următor sau **NIL/NULL** dacă nu există un element următor în listă. Lista are cel puţin două elemente. Ştiind că variabila **p** indică către primul nod din listă iar variabila **u** indică către al doilea nod, care este secvenţa corectă prin care se inserează între cele două noduri **p** si **u** un nod indicat prin variabila **q**?

a.	`q^.urm:=u; u^.urm:=p;`	**a.**	`q->urm=u; u->urm=p;`
b.	`u^.urm:=p; p^.urm:=q;`	**b.**	`q->urm=p; u->urm=q;`
c.	`q^.urm:=u; p^.urm:=q;`	**c.**	`q->urm=u; p->urm=q;`
d.	`q^.urm:=p; u^.urm:=q;`	**d.**	`u->urm=p; p->urm=q;`

43. Se consideră o listă liniară simplu înlănţuită, cu cel puţin două noduri. Fiecare nod reţine în câmpul **info** o valoare numerică iar în câmpul **adr** adresa următorului nod din listă. Dacă **p** este adresa unui nod din listă atunci **p** şi **p->adr** conţin aceeaşi informaţie în câmpul **info** dacă şi numai dacă:

a. `p^.adr = p`
b. `p = p^.info`
c. `p^.info = p^.adr^.info`
d. `p.info = p^.adr.info`

a. `p->adr == p`
b. `p == p->info`
c. `p->info == p->adr->info`
d. `p.info == p->adr.info`

44. O listă liniară simplu înlănţuită alocată dinamic, în care fiecare element memorează în câmpul **nr** un număr întreg, iar în câmpul **urm** adresa elementului următor din listă, conţine exact trei elemente ale căror adrese sunt memorate în variabilele **p**, **q** şi **r**. Ştiind că :

`q^.nr=3`, `p^.nr=5`, `r^.nr=8`, `p^.urm<>NIL` şi `r^.urm=q`

`q->nr=3`, `p->nr=5`, `r->nr=8`, `p->urm!=NULL` şi `r->urm=q`,

care este ordinea numerelor din listă?

a. `8, 3, 5` **b.** `5, 8, 3` **c.** `3, 8, 5` **d.** `5, 3, 8`

45. Într-o listă simplu înlănţuită, fiecare nod memorează în câmpul **adr** adresa nodului următor. Lista conţine **4** noduri, iar variabila **p** memorează adresa primului nod a listei. Pentru eliminarea celui de-al treilea nod din listă vom utiliza instrucţiunea:

a. `p^.adr:=p^.adr^.adr;`
b. `p^.adr^.adr:=NIL;`
c. `p:=p^.adr^.adr;`
d. `p^.adr^.adr:= p^.adr^.adr^.adr;`

a. `p->adr=p->adr->adr;`
b. `p->adr->adr=NULL;`
c. `p=p->adr->adr;`
d. `p->adr->adr= p->adr->adr->adr;`

46. Fiecare nod al unei liste liniare simplu înlănţuite, alocată dinamic, reţine numele format din maximum **20** de caractere şi vârsta unui elev, precum şi adresa nodului următor. Care este declararea corectă a structurii de date corespunzătoare?

a.
```
typedef pnod=^nod;
        nod = record
         nume:string[21];
         v:integer; urm:pnod
        end;
```
b.
```
typedef pnod=nod;
        nod = record
          nume:string[20];
          v:integer;urm:pnod
        end;
```

a.
```
struct nod
{  char nume[21];
   int v;
   nod urm;
}
```
b.
```
struct nod
{ char nume[21];
  nod*urm;
}
```

c.
```
typedef pnod=^nod;
        nod = record
          nume:string[20];
          v:integer; urm:nod
        end;
```

d.
```
typedef pnod=^nod;
        nod = record
          nume:string[20];
          v:integer;urm:pnod
        end;
```

c.
```
struct nod
{ char nume[20];
  int *v; nod*urm;
}
```

d.
```
struct nod
{ char nume[21];
  int v; nod*urm;
}
```

47. Lista simplu înlanţuită alocată dinamic din secvenţa alăturată are **10** noduri ce reţin in câmpul **urm** adresa nodului următor sau **NIL/NULL** dacă nu există un element următor, iar în câmpul **info** câte o valoare întreagă din intervalul **[1,10]**. Valorile sunt memorate în ordine crescătoare, astfel: primul nod conţine valoarea **1**, cel de-al doilea **2**, etc. Dacă **p** reţine iniţial adresa primului element al listei, ce valoare se afişează?

```
s:=0;
while p<>nil do begin
  s:=s+p^.info;
  p:=p^.urm^.urm end;
write(s)
```

```
s=0;
while (p<>NULL)  {
  s=s+p->info;
  p=p->urm->urm; }
printf("%d",s); /  cout<<s;
```

a. 30 **b.** 55 **c.** 10 **d.** 25

48. Dacă **p** este adresa primului element al unei liste simplu înlănţuite alocată dinamic în care fiecare element reţine în câmpul **urm** adresa elementului următor, atunci secvenţa alăturată va atribui variabilei **p** adresa:

```
while p^.urm<>nil do
    p=p^.urm;
```

```
while(p->urm!=0)
    p=p->urm;
```

a. antepenultimului nod **b.** penultimului nod **c.** ultimului nod **d.** **NULL**

49. Se consideră o listă liniară simplu înlănţuită ale cărei noduri reţin în câmpul **inf** informaţia fiecărui nod, iar în câmpul **urm** adresa nodului următor. Lista are cel puţin patru elemente. Ştiind că **p** reţine adresa celui de-al doilea nod din listă, ce instrucţiune trebuie executată pentru a afişa informaţia memorată în al patrulea nod?

a.
```
write(p^.urm^.inf^.urm)
```
```
printf("%d",p->urm->inf->urm);
/ cout<<p->urm->inf->urm;
```

b.
```
write(p^.urm^.inf)
```
```
printf("%d",p->urm->inf);
/ cout<<p->urm->inf;
```

c.
```
write(p^.urm^.urm^.inf)
```
```
printf("%d",p->urm->urm->inf);
 / cout<<p->urm->urm->inf;
```

d.
```
write(p^.urm^.urm^.urm^.inf)
```
```
printf("%d",p->urm->urm->urm->inf);
 / cout<<p->urm->urm->urm->inf;
```

50. Într-o listă simplu înlanţuită cu cel puţin 2 elemente fiecare element memorează în câmpul **next** adresa elementului următor din listă iar în câmpul **data** un număr întreg.

Ştiind că **prim** reprezintă adresa primului element din listă, ce realizează următoarea secvenţă de program?

```
p:=prim;
while p^.next<> nil do
    p:=p^.next;
write(p^.data);
```

```
p=prim;
while(p->next!=NULL)
    p=p->next;
cout<<p->data;
 / printf("%d",p->data);
```

a. afişează informaţia din primul nod al listei
b. afişează informatia din penultimul nod al listei
c. afişează informatia din ultimul nod al listei
d. afişează informatia din toate nodurile listei

51. Se consideră o listă circulară dublu înlănţuită ale cărei noduri reţin în câmpul **st** adresa nodului anterior iar în câmpul **dr** adresa nodului următor din listă. Lista are cel puţin două elemente. Ştiind că **p** reţine adresa unui nod din listă, care este numărul de noduri din listă, astfel încât relaţia

```
p^.st^.st = p^.dr
```

```
p->st->st == p->dr
```

să fie adevărată?

a. 5 **b.** 3 **c.** 2 **d.** 4

52. Într-o listă liniară simplu înlănţuită, circulară, nevidă, fiecare element reţine în câmpul **urm** adresa elementului următor din listă, iar în câmpul **info** (de tip **integer/int**), informaţia utilă.

Dacă **p** reţine adresa unui element al listei, ce condiţie trebuie testată în instrucţiunea **while** astfel încât să se afişeze valorile reţinute în câmpul **info** pentru toate elementele listei, o singură dată?

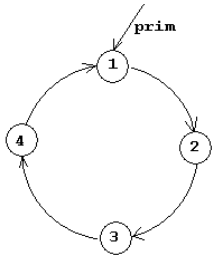
```
write(p^.info,' ');
q:=p^.urm;
while … do
 begin
  write(q^.info,' ');
  q:=q^.urm
 end;
```

a. q<>p^.urm **b.** q=p
c. q^.urm<>p **d.** q<>p

```
cout<<p->info<<" ";
     /printf("%d",p->info);
q=p->urm;
while( … )
{
  cout<<q->info<<" ";
       /printf("%d ",q->info);
  q=q->urm;
}
```

a. q!=p->urm **b.** q==p
c. q->urm !=p **d.** q!=p

53. Se consideră lista circulară simplu înlănţuită din figura alăturată în care fiecare element memorează în câmpul **nr** un număr natural şi în câmpul **urm** adresa elementului următor din listă.



Pentru variabila **prim** din figură, stabiliţi câte **treceri** sunt necesare pentru ca toate elementele din listă să ajungă egale.

Definim prin **trecere** prelucrarea dată de secvenţa următoare:

```
p:=prim;
repeat
if p^.nr < p^.urm^.nr then
           p^.nr:=p^.nr+1;
p:=p^.urm
until p=prim;
```

```
p=prim;
do{ if(p->nr < p->urm->nr)
         p->nr = p->nr +1;
    p=p->urm;
  } while(p!=prim);
```

a. 5 **b.** 2 **c.** 3 **d.** 4

54. Într-o listă circulară simplu înlănţuită alocată dinamic cu cel puţin un element, fiecare element reţine în câmpul **nr** un număr întreg şi în câmpul **urm** adresa următorului element din listă.

Ştiind că variabila **p** reţine adresa unui element din listă şi variabila **t** este de acelaşi tip cu variabila **p**, stabiliţi care dintre următoarele secvenţe afişează toate valorile memorate în elementele listei, fiecare valoare fiind afişată exact o dată:

a.
```
t:=p;
while(t^.urm<>p)       do
begin
 write(t^.nr,'        ');
t:=t^.urm  end
```

b.
```
t:=p;
repeat
   write(t^.nr,'      ');
t:=t^.urm
until(t=p)
```

c.
```
t:=p;
while(t<>p) do begin
 write(t^.nr,'        ');
t:=t^.urm
end
```

d.
```
t:=p^.urm;
repeat
   write(t^.nr,'      ');
t:=t^.urm
until(t=p)
```

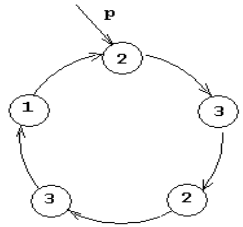
a.
```
t=p;
while(t->urm!=p){
  cout<<t->nr<<" ";
  / printf("%d",t->nr);
 t=t->urm; }
```

b.
```
t=p;
do{ cout<<t->nr<<" ";
          / printf("%d ",t->nr);
    t=t->urm;
}while(t!=p);
```

c.
```
t=p;
while(t!=p){
 cout<<t->nr<<",";
 / printf("%d",t->nr);
 t=t->urm; }
```

d.
```
t=p->urm;
do{ cout<<t->nr<<",";
       / printf("%d",t->nr);
    t=t->urm;
}while(t!=p);
```

55. În lista circulară simplu înlănţuită alăturată, fiecare element memorează în câmpul **nr** un număr natural şi în câmpul **urm** adresa elementului următor din listă. Dacă **p** este adresa nodului de plecare, atunci, după executarea secvenţei




```
while p^.nr>0 do begin
  p^.nr:=p^.nr-1; p:=p^.urm
                  end;
```

```
while(p->nr>0)
 {p->nr=p->nr-1;
  p=p->urm;}
```

conţinutul listei, citit de la adresa de plecare, va fi:

a. 0 1 0 2 0 **b.** 1 2 1 2 0 **c.** 0 1 1 2 0 **d.** 0 1 0 1 0

56. Se consideră o lista circulară cu **8** elemente numerotate cu **1,2,3,4....8**. La fiecare pas se elimină fiecare al treilea element al listei, numărarea continuându-se începând cu succesorul elementului eliminat, până când lista va mai conţine un singur element. Care va fi numărul de ordine al elementului rămas, dacă se începe eliminarea cu al treilea element?

a. 2 **b.** 7 **c.** 3 **d.** 4

57. Într-o listă circulară simplu înlănţuită, cu cel puţin un element, fiecare nod reţine în câmpul **adr** adresa elementului următor din listă. Dacă **p** este o variabilă care reţine adresa primului element din listă, iar **q** este o variabilă care poate să reţină adresa unui element din listă, care dintre următoarele secvenţe de instrucţiuni calculează în variabila **nr**, de tip **int**, numărul de elemente ale listei?

a.
```
nr:=0;q:=p;
while q<>p do begin
    nr:=nr+1;q:=q^.adr end;
```

b.
```
nr:=0;q:=p;
repeat nr:=nr+1;q:=q^.adr
until q=p;
```

c.
```
nr:=0;q:=p;
repeat nr:=nr+1;q:=p^.adr
until q=p;
```

d.
```
nr:=0;q:=p;
while p<>q do begin
    nr:=nr+1;p:=p^.adr end;
```

a.
```
nr=0;q=p;
while(q!=p)
     {nr++;q=q->adr;}
```

b.
```
nr=0;q=p;
do{nr++;
  q=q->adr;}while(q!=p);
```

c.
```
nr=0;q=p;
do{nr++;
  q=p->adr;}while(q!=p);
```

d.
```
nr=0;q=p;
while(p!=q)
    {nr++;p=p->adr;}
```

58. Într-o listă circulară simplu înlănţuită, **p** este adresa unui nod din listă şi câmpul **next** memorează pentru fiecare nod, adresa nodului următor din listă. Pentru a număra elementele listei vom scrie secvenţa (variabila **q** este de acelaşi tip cu variabila **p**)

VARIANTA PASCAL

a.
```
q:=p; k:=1;while(q^.next<>p)do begin k++;q:=q^.next;end;
```
b.
```
q:=p; k:=1;repeat q:=q^.next;k++; until(q<>p);
```
c.
```
q:=p; k:=1;while(q<>p) begin k++;q:=q^.next;end;
```
d.
```
k:=0;repeat p:=p^.next;k++; until p=Nil;
```

VARIANTA C/C++

a.
```
q=p; k=1;while(q->next!=p) {k++;q=q->next;}
```
b.
```
q=p; k=1;do {q=q->next;k++;} while (q==p);
```
c.
```
q=p; k=1;while(q!=p) {k++;q=q->next;}
```
d.
```
k=0;do {p=p->next;k++;} while (p!=NULL);
```

59. Într-o listă circulară simplu înlănţuită, fiecare nod al listei reţine în câmpul **next** adresa nodului următor din listă. Dacă variabilele **p** şi **q** ce memorează adresele a două noduri distincte din listă, verifică relaţiile:

`p^.next^.next=q`	`p->next->next==q`
`q^.next^.next=p`	`q->next->next==p`

atunci numărul total de noduri din listă este:

a. 5 **b.** 4 **c.** 2 **d.** 3

60. Variabila **p** reţine adresa unui element oarecare al unei liste circulare nevide alocată dinamic, în care fiecare element memorează în câmpul **nr** un număr întreg, iar în câmpul **urm** adresa elementului următor. Care dintre următoarele variante tipăreşte toate elementele listei?

a.
```
q:=p;
repeat
       write(q^.nr);
       q:=q^.urm
until q=p
```
b.
```
q:=p;
while q^.urm<>p do
 begin write(q^.nr);
    q:=q^.urm
end
```
c.
```
q:=p;
while q<>p do
begin
  write(q^.nr); q:=q^.urm
end
```
d.
```
q:=p^.urm;
while q<>p do
begin
   write(q^.nr); q:=q^.urm
end
```

a.
```
q=p;
do { printf("%d",q->nr);
     / cout<<q->nr;
    q=q->urm;}
while(q!=p);
```
b.
```
q=p;
while(q->urm!=p)
{ printf("%d",q->nr);
          / cout<<q->nr;
  q=q->urm;}
```
c.
```
q=p;
while(q!=p)
{printf("%d",q->nr);
       / cout<<q->nr;
 q=q->urm;}
```
d.
```
q=p->urm;
while(q!=p)
 { printf("%d",q->nr);
      / cout<<q->nr;
   q=q->urm;}
```

61. Într-o listă circulară simplu înlănţuită fiecare element reţine în câmpul **adr** adresa elementului următor din listă. Dacă **p** reprezintă adresa unui element din listă, atunci stabiliţi care dintre următoarele expresii are valoarea **1** dacă şi numai dacă lista conţine cel mult două noduri.

a. `p^.adr=p`	**a.** `p->adr==p`
b. `p^.adr^.adr=nil`	**b.** `p->adr->adr==NULL`
c. `p^.adr^.adr=p`	**c.** `p->adr->adr==p`
d. `p^.adr<>nil`	**d.** `p->adr!=NULL`

62. Se consideră că variabilele **p** şi **q** memorează adresa primului, respectiv ultimului element al unei liste liniare nevide dublu înlănţuite. Elementele listei reţin în câmpul **urm** adresa elementului următor iar în câmpul **prec** adresa elementului anterior. Stabiliţi care este numărul de noduri din listă dacă **p^.urm^.urm** | **p->urm->urm** şi **q^.prec^.prec** | **q->prec->prec** indică acelaşi nod al listei.

a. 4 **b.** 5 **c.** 3 **d.** 2

63. Într-o listă dublu înlănţuită cu exact 4 elemente, fiecare element reţine în câmpurile **adp** şi **adu** adresa elementului precedent şi respectiv adresa elementului următor din listă, iar în câmpul **info** (de tip **integer/int**) informaţia utilă. Dacă **p** reţine adresa primului element din listă şi **p^.info** (Pascal), respectiv **p->info** (C/C++), reţine valoarea **1**, **p^.adu^.info** (Pascal), respectiv **p->adu->info** (C/C++), reţine valoarea **2** şi lista are forma: **1** ⇆ **2** ⇆ **3** ⇆ **4**, ce afişează instrucţiunea următoare?

Pascal	`writeln(p^.adu^.adu^.adp^.adu^.info);`
C/C++	`cout<<p->adu->adu->adp->adu->info;` `/printf("%d",p->adu->adu->adp->adu->info);`

a. 2 **b.** 4 **c.** 3 **d.** 1

64. Într-o listă dublu înlănţuită cu exact **5** elemente, fiecare element reţine în câmpurile **adp** şi **adu** adresa elementului precedent şi respectiv adresa elementului următor din listă, iar în câmpul **info** (de tip **integer/int**) informaţia utilă. Dacă **p** reţine adresa celui de-al treilea element din listă, **p^.info** (Pascal), respectiv **p->info** (C/C++), reţine valoarea **3**, **p^.adu^.info** (Pascal), respectiv **p->adu->info** (C/C++), reţine valoarea **4** şi lista are forma: **1** ⇆ **2** ⇆ **3** ⇆ **4** ⇆ **5**, ce afişează instrucţiunea următoare?

Pascal	`write(p^.adp^.adp^.adu^.info);`
C/C++	`cout<<p->adp->adp->adu->info;` `/ printf("%d", p->adp->adp->adu->info);`

a. 5 **b.** 3 **c.** 1 **d.** 2

65. Într-o listă dublu înlănţuită alocată dinamic, cu cel puţin patru elemente, fiecare element reţine în câmpul **urm**, respectiv **pred**, adresa elementului următor, respectiv precedent, din listă. Dacă variabilele **p**, **q** şi **r** reţin adresele a trei elemente din listă astfel încât:

r^.urm^.urm=q şi **q^.pred=p** | **r->urm->urm==q** şi **q->pred==p**

atunci este adevărată condiţia:

a. **q^.pred^.pred=p**	**a.** **q->pred->pred==p**
b. **r^.urm^.pred=q**	**b.** **r->urm->pred==q**
c. **p^.urm^.pred=r**	**c.** **p->urm->pred==r**
d. **p^.urm=q**	**d.** **p->urm==q**

66. Într-o listă liniară dublu înlănţuită cu cel puţin **3** elemente, fiecare element memorează în câmpul **urm** adresa elementului următor în listă şi în câmpul **prec** adresa elementului precedent în listă, iar în variabila **p** adresa primului element din listă şi în variabila **q** adresa ultimului element din listă. Dacă valoarea expresiei

Varianta Pascal	**p^.urm^.urm^.urm=q^.prec^.prec**
Varianta C/C++	**p->urm->urm->urm==q->prec->prec**

este **true/1**, câte elemente are lista?

a. 7 **b.** 6 **c.** 5 **d.** 3

67. Într-o listă liniară dublu înlănţuită cu cel puţin **2** elemente, fiecare element memorează în câmpul **urm** adresa elementului următor în listă şi în câmpul **prec** adresa elementului precedent în listă, iar în variabila **p** adresa primului element din listă şi în variabila **q** adresa ultimului element din listă. Dacă valoarea expresiei

p^.urm^.prec^.urm=q^.prec | **p->urm->prec->urm==q->prec**

este **true/1**, câte elemente are lista?

a. 2 **b.** 4 **c.** 3 **d.** 5

68. Într-o lista dublu înlănţuită care începe cu elementul memorat la adresa **p** şi conţine cel puţin **4** elemente, fiecare element reţine în câmpul **urm** adresa elementului următor, în câmpul **pre** adresa elementului precedent, iar în câmpul **inf** o valoare întreagă. Care dintre următoarele variante tipăreşte valoarea celui de-al treilea element al listei?

Varianta Pascal	**a.** **Writeln (p^.urm^.urm^.pre^.inf)**
	b. **Writeln (p^.urm^.urm^.urm^.pre^.inf)**
	c. **Writeln(p^.urm^.urm^.urm)**
	d. **Writeln (p^.urm^.urm)**

	a. `printf("%d", p->urm->urm->pre->inf);` `/ cout<<p->urm->urm->pre->inf;`
Varianta	**b.** `printf("%d", p->urm->urm->urm->pre->inf);` `/ cout<<p->urm->urm->urm->pre->inf;`
C/C++	**c.** `printf("%d", p->urm->urm->urm);` `/ cout<<p->urm->urm->urm;`
	d. `printf("%d", p->urm->urm); / cout<<p->urm->urm;`

69. Într-o listă dublu înlănţuită nevidă, fiecare element reţine în câmpul **urm** adresa elementului următor din listă iar în câmpul **pre** adresa elementului precedent din listă. Variabilele **r, q** şi **p**, sunt de tip pointer/referinţă astfel declarate încât pot reţine adresa oricărui element al listei. Ştiind că variabila **r** reţine adresa unui element oarecare din listă, cu ce trebuie înlocuite cele trei puncte din secvenţa alăturată astfel încât ea sa calculeze corect în variabila **nr** declarată de tip **longint**/**long**, numărul de elemente din această listă. Variabilele **nrs** şi **nrd** sunt tot de tip **longint**/**long**.

```
q := r; p := r;
nrs :=0; nrd :=0;
while p<>nil
 begin
  nrd := nrd + 1;
  p := p->urm
 end;
while q<>nil
 begin
  nrs := nrs + 1;
  q := q->pre
 end;
nr = …;
```

```
q = r; p = r;
nrs = 0; nrd = 0;
while (p!=NULL)
{ nrd = nrd + 1;
  p = p->urm;  };
while (q!=NULL)
{ nrs = nrs + 1;
  q = q->pre;  };
nr = …;
```

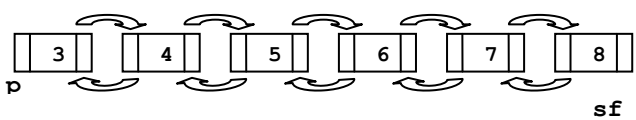
a. `nrs + nrd – 1`

b. `nrs + nrd + 1`

c. `nrs + nrd`

d. `nrs + nrd – 2`

70. Se consideră lista dublu înlănţuită din desenul următor :



Fiecare element reţine în câmpul **inf** un număr natural, în câmpurile **st** şi **dr** reţine adresa nodului precedent şi respectiv adresa nodului următor din listă. Variabilele globale **p** şi **sf** reţin adresele primului şi respectiv ultimului element din listă. O variabilă ce reţine adresa unui element este de tip **nod**. Care va fi conţinutul listei la o parcurgere de la stânga la dreapta după apelul subprogramului **sub**?

```
procedure sub;
var man,q:ref;
begin
 man:=sf;
 while man^.inf>sf^.inf div 2 do
    man:=man^.st;
 q:=man;
 man^.st^.dr:= q^.dr;
 q^.dr^.st:= man^.st;
 dispose(q)
end;
```

```
void sub ()
{ nod *man=sf;
  while(man->inf>sf->inf/2)
     man=man->st;
  nod *q=man;
  man->st->dr=q->dr;
  q->dr->st=man->st;
  delete q;
 }
```

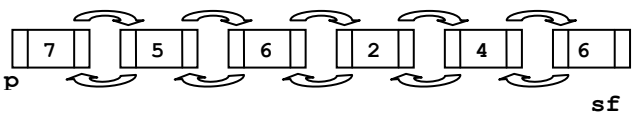
a. 3 5 6 7 8

b. 4 5 6 7 8

c. 3 4 5 6 7 8

d. 3 4 6 7 8

71. Se consideră lista dublu înlănţuită din desenul următor:



Fiecare element reţine în câmpul **inf** un număr natural, în câmpurile **st** şi **dr** reţine adresa nodului precedent şi respectiv adresa nodului următor din listă. Variabilele globale **p** şi **sf** reţin adresele primului şi respectiv ultimului element din listă. O variabilă ce reţine adresa unui element este de tip **nod**.

Care va fi conţinutul listei la o parcurgere de la stânga la dreapta după apelul subprogramului **sub**?

```
procedure sub;
var man,q:ref;
begin
 man:=sf;
 while man^.inf>=sf^.inf do
    man:=man^.st;
 q:=man;
 man^.st^.dr:= q^.dr;
 q^.dr^.st:= man^.st;
 dispose(q)
end;
```

```
void sub()
{
 nod*man=sf;
 while(man->inf>=sf->inf)
    man=man->st;
 nod*q=man;
 man->st->dr=q->dr;
 q->dr->st=man->st;
 delete q;
}
```

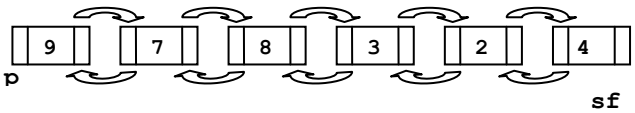
a. 7 5 6 2 4 6

b. 7 5 6 2 6

c. 7 5 6 4 6

d. 7 5 6 2 4

72. Se consideră lista dublu înlănţuită din desenul următor:



Fiecare element reţine în câmpul **inf** un număr natural, iar în câmpurile **st** şi **dr** reţine adresa nodului precedent şi respectiv adresa nodului următor din listă. Variabilele globale **p** şi **sf** reţin adresele primului şi respectiv ultimului element din listă. O variabilă ce reţine adresa unui element este de tip **nod**. Care va fi conţinutul listei la o parcurgere de la stânga la dreapta după apelul subprogramului **sub**?

```
procedure sub;
var man,q:ref;
begin
 man:=sf;
 while man^.inf<=sf^.inf do
    man:=man^.st;
 q:=man;
 man^.st^.dr:= q^.dr;
 q^.dr^.st:= man^.st;  dispose(q)
end;
```

```
void sub()
{
 nod*man=sf;
 while(man->inf<=sf->inf)
    man=man->st;
 nod*q=man;
 man->st->dr=q->dr;
 q->dr->st=man->st;
 delete q;
}
```

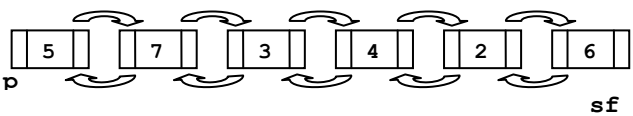
a. 9 7 3 2 4

b. 9 7 8 2 4

c. 9 7 8 3 2 4

d. 9 8 3 2 4

73. Se consideră lista dublu înlănţuită din desenul următor:



Fiecare element reţine în câmpul **inf** un număr natural, iar în câmpurile **st** şi **dr** reţine adresa elementului precedent şi respectiv adresa elementului următor din listă. Variabilele globale **p** şi **sf** reţin adresele primului şi respectiv ultimului element din listă. O variabilă ce reţine adresa unui element este de tip **ref**. Care va fi conţinutul listei la o parcurgere de la stânga la dreapta după apelul subprogramului **sub**?

```
procedure sub;
var man,q:ref;
begin
 man:=sf^.st;
 while man^.inf<sf^.inf do
    man:=man^.st;
 q:=man;
 man^.st^.dr:= q^.dr;
 q^.dr^.st:= man^.st;  dispose(q)
end;
```

```
void sub()
{
 nod*man=sf->st;
 while(man->inf<sf->inf)
    man=man->st;
 nod*q=man;
 man->st->dr=q->dr;
 q->dr->st=man->st;
 delete q;
}
```

a. 5 3 4 2 6 **b.** 5 7 4 2 6

c. 7 3 4 2 6 **d.** 5 7 3 4 2 6

74. Într-o listă dublu înlănţuită, fiecare nod al listei reţine în câmpurile **urm** şi **prec** adresa nodului următor şi respectiv a nodului precedent din listă. Dacă variabilele **p** şi **q**, ce memorează adresele a două noduri distincte din listă, verifică relaţia

`p^.urm^.urm=q^.prec^.prec`	`p->urm->urm==q->prec->prec`

atunci între nodurile de la adresele **p** şi **q** există alte:

a. 3 noduri **b.** 5 noduri **c.** 2 noduri **d.** 4 noduri

6.2. Probleme

1. Se consideră următoarele definiţii:

```
type pnod=^nod;
     nod=record
        nr:longint; adr:pnod
      end;
```

```
struct nod
      { long nr;
        nod *adr;
      };
```

a) Scrieţi definiţia completă a subprogramului **sub** care, prin intermediul unui parametru **p** primeşte adresa primului element dintr-o listă simplu înlănţuită alocată dinamic, iar prin parametru **k** un număr natural cu cel mult opt cifre; subprogramul returnează numărul elementelor din listă care conţin valori mai mici sau egale cu k.

Exemplu: dacă lista contine numerele: **12, 5, 30, 18, 47, 8** si **k=25** subprogramul returnează valoarea **4**.

b) Se consideră o listă liniară simplu înlănţuită alocată dinamic cu elemente de tipul **nod**. Variabila **prim** reţine adresa primului nod al listei. Scrieţi declarările de variabile necesare şi secvenţa de program Pascal/C/C++ în care se citesc două numere naturale **a** şi **b** (fiecare fiind format din cel mult opt cifre şi **a<b**) şi care determină şi afişează pe ecran numărul valorilor din listă ce aparţin intervalului **(a,b]**, folosind apeluri ale subprogramului de la punctul **a)**.

Exemplu: dacă lista contine numerele: **6, 9, 7, 3, 5, 8** şi **a=4**, **b=7** se va afisa: **3**

2. Se consideră subprogramele:

creare - construieşte o listă simplu înlănţuită alocată dinamic ce memorează în fiecare nod, în ordine, câte o literă a unui cuvânt **c** transmis ca parametru; adresa primului nod al listei este returnată printr-un al doilea parametru, **p**;

listare – afişează pe ecran, în ordine, literă cu literă, conţinutul listei, adresa primului element fiind transmisă prin intermediul parametrului **p**.

a) Scrieţi declarările ce definesc lista şi antetul corect pentru fiecare dintre subprogramele **creare** şi **listare**.

b) Scrieţi programul Pascal/C/C++ (cu excepţia definiţiilor subprogramelor **creare** şi **listare**), program care citeşte de la tastatură un cuvânt format din cel mult 30 de litere mari, formează o listă ce reţine literele cuvântului citit cu ajutorul subprogramului **creare**, elimină din listă un număr minim de noduri astfel încât să nu existe noduri succesive (vecine) ce reţin o aceeaşi literă, iar în final afişează conţinutul listei rămase folosind subprogramul **listare**.

De exemplu, pentru cuvântul **STTANDD**, se afişează în final **STAND**.

3. Subprogramul **minim**, primeşte printr-un parametru **p** adresa unui element oarecare al unei liste circulare simplu înlănţuită, nevidă, alocată dinamic, care conţine numere întregi cu cel mult **3** cifre fiecare şi returnează valoarea elementului minim din listă.

a) Scrieţi definiţia tipurilor de date utilizate.

b) Scrieţi definiţia completă a subprogramului **minim**.

4. Scrieţi declarările de tipuri necesare şi definiţia completă a unei funcţii **elim2**, cu un parametru, funcţie care primeşte prin intermediul parametrului **p** adresa primului element al unei liste simplu înlănţuite alocată dinamic ce reţine ca informaţii utile valori naturale de cel mult **6** cifre fiecare;
dacă lista are cel puţin două elemente, elimină al doilea element al listei şi returnează valoarea ce fusese reţinută de elementul eliminat, iar dacă lista nu are cel puţin două elemente, returnează valoarea **-1** .

5. Se defineşte un tip de listă simplu înlănţuită alocată dinamic ce memorează în câmpul **num** un număr întreg de cel mult **4** cifre şi în cîmpul **next** adresa elementului următor din listă. Considerăm definite următoarele trei subprograme:

- **inserare** – cu doi parametri: **prim** ce memorează adresa primului element al listei şi **nr** un număr întreg de cel mult **4** cifre; subprogramul realizează adăugarea înaintea primului nod din listă a unui nod nou, având ca informaţie utilă numărul **nr;**
- **numarare** – cu doi parametri: **prim** ce memorează adresa primului element al listei şi **nr** un număr întreg format din cel mult **4** cifre şi returnează numărul elementelor ce reţin în câmpul **num** valoarea **nr**.

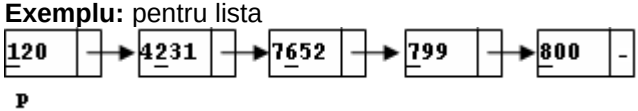
a) Scrieţi declararea tipurilor de date necesare pentru definirea listei şi doar antetul fiecăruia dintre cele două subprograme.

b) Scrieţi un subprogram care primeşte prin intermediul unui parametru adresa primului element al unei liste nevide de tipul mai sus indicat, şi,

apelând subprogramele definite anterior, creează o listă simplu înlănţuită de acelaşi tip cu prima, listă care să reţină doar numerele ce apar de cel puţin două ori în prima listă. În lista nou creată, fiecare dintre aceste numere va fi inserat o singură dată. Subprogramul va returna adresa primului element al listei nou create.

6. O listă liniară simplu înlănţuită, alocată dinamic, reţine în câmpul **info** al fiecărui element câte un număr natural din intervalul **[100, 10000]**, iar în câmpul **adr**, adresa elementului următor din listă. Să se scrie declarările de tipuri necesare definirii listei, precum şi definiţia completă a subprogramului **scrie**, care are un parametru, **p**, prin care primeşte adresa de început a listei şi afişează pe ecran, separate printr-un spaţiu, acele numere memorate în listă care au cifra sutelor pară.

Exemplu: pentru lista



se afişează **4231 7652 800**

7. Se consideră definite subprogramele:

- **creare**, care citeşte de la tastatură mai multe cuvinte de cel mult **30** de litere fiecare şi creează o listă simplu înlănţuită ce conţine în fiecare nod al său, în câmpul **info**, câte un cuvânt dintre cele citite, iar în câmpul **adr**, adresa următorului nod din listă. Subprogramul are un singur parametru, **p** reprezentând adresa de început a listei nou create;

-cautare, care caută în lista creată un anumit cuvânt; subprogramul are ca parametri variabila **p** prin care se transmite adresa de început a listei şi variabila **x** prin care se transmite cuvântul căutat; subprogramul returnează adresa nodului care conţine cuvântul căutat, sau **NULL/nil** în cazul în care acest cuvânt nu este găsit în listă.

a) Scrieţi declarările de date ce definesc lista, antetul subprogramului **creare** şi definiţia completă a subprogramului **cautare**.

b) Scrieţi programul **PASCAL/C/C++** care construieşte o listă cu ajutorul subprogramului **creare** şi caută în lista creată cuvântul **bacalaureat** folosind subprogramul **cautare**. Programul afişează pe ecran un mesaj dacă acest cuvânt a fost găsit sau nu în lista creată.

8. Se introduce de la tastatură o frază de maximum **256** de caractere, ale cărei cuvinte sunt scrise cu majuscule. Scrieţi programul care memorează într-o listă simplu înlănţuită, în ordine alfabetică, majusculele care apar de mai multe ori (cel puţin de două ori) în frază. Fiecare nod al listei va memora o majusculă din text şi numărul de apariţii al respectivei majuscule în fraza dată. Afişaţi lista astfel formată, pe fiecare rând de ecran se va afişa o literă şi numărul ei de apariţii. În cazul în care fraza are caractere distincte se va afişa mesajul **Listă vidă**.

Exemplu: pentru textul introdus de la tastatură: **SUCCES LA EXAMEN**

Lista afişată va fi:
A 2
C 2
E 3
S 2

9. Fie o listă liniară simplu înlănţuită, în care fiecare nod reţine în câmpul **inf** un caracter şi în câmpul **next** adresa nodului următor. Se consideră subprogramele:
- **cre**, care creează o listă liniară simplu înlănţuită ce conţine toate literele unui cuvânt **cuv,** în ordinea în care acestea apar în cuvânt. Subprogramul are parametrii şirul de caractere **cuv** (cu cel mult **50** de caractere) şi adresa **p** de început al listei;
- **sterg**, care elimină din listă toate nodurile ce memorează caractere egale cu un caracter **c**. Subprogramul are parametrii adresa **p** de început al listei şi caracterul **c**;
- **lungime**, care determină şi returnează numărul de noduri ale listei; adresa de început a listei este dată prin singurul parametru **p**.

Un nod al listei este de tipul **nod**.

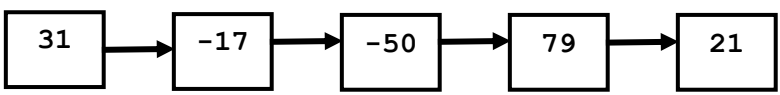
a) Scrieţi antetul corect pentru fiecare din cele 3 subprograme.

b) Scrieţi declarările de variabile şi programul principal care citeşte de la tastatură un cuvânt de cel mult **50** de litere şi afişează pe ecran numărul de vocale ale cuvântului, utilizând doar apeluri ale subprogramelor de la a).
De exemplu, dacă se citeşte cuvântul **bacalaureat**, pe ecran se va afişa **6**.

10. Se consideră o listă simplu înlănţuită (cu cel puţin două elemente) în care fiecare element reţine în câmpul **info** un număr întreg nenul format din cel mult patru cifre, iar în câmpul **adr** adresa următorului element din listă.

Scrieţi definiţiile de date şi definiţia completă a subprogramului **S1** care are ca parametru adresa **p** a primului element al listei şi care returnează numărul schimbărilor de semn obţinute în urma parcurgerii listei.
(O schimbare de semn apare când valorile conţinute de două elemente succesive din listă au semne diferite.)

Exemplu: **p**



Se obţine: **k=2**

11. Se consideră o listă simplu înlănţuită (cu cel puţin două elemente) în care fiecare element reţine în câmpul **info** un număr real, iar în câmpul **adr** adresa următorului element din listă.

Scrieţi definiţiile de date şi definiţia completă a subprogramului **S1** care are ca parametru adresa **p** a primului element al listei şi care determină

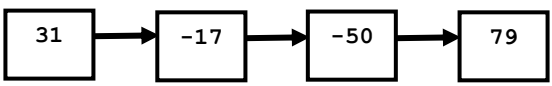
afişarea pe ecran a mesajului : ″**elementele listei sunt ordonate strict descrescător**″ dacă elementele listei sunt ordonate strict descrescător; ″**elementele listei sunt ordonate strict crescător**″ dacă elementele listei sunt ordonate strict crescător; ″**elementele listei nu sunt ordonate strict**″ în celelalte cazuri.

12. Se consideră o listă simplu înlănţuită (cu cel puţin două elemente) în care fiecare element reţine în câmpul **info** un număr întreg nenul format din cel mult patru cifre, iar în câmpul **adr** adresa următorului element din listă.

Scrieţi definiţiile de date şi definiţia completă a subprogramului **S1** care are ca parametru adresa **p** a primului element al listei şi care returnează adresa elementului din mijlocul listei. În cazul în care numărul de elemente din listă este par, se va returna adresa ultimului element din prima jumătate a listei.

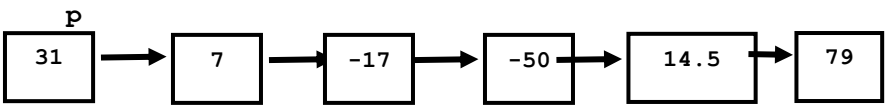
13. Se consideră o listă simplu înlănţuită (cu cel puţin două elemente) în care fiecare element reţine în câmpul **info** un număr real nenul, iar în câmpul **adr** adresa următorului element din listă. Scrieţi definiţiile de date şi definiţia completă a subprogramului **S1** care are ca parametru adresa **p** a primului element al listei şi care modifică lista prin inserarea între oricare două elemente de semne contrare a unui nou element a cărui valoare este egală cu media aritmetică a celor două elemente.

Ex. **p**



Se obţine :

p



14. O listă liniară simplu înlănţuită, reţine în fiecare nod, o valoare întreagă în câmpul **info** şi adresa nodului următor, în câmpul **urm**.. Scrieţi în **Pascal/C/C++**, subprogramul **Insereaza**, care inserează în listă un nod. Subprogramul are ca parametru adresa nodului după care se va insera noul nod. Câmpul **info** al nodului inserat, reţine partea întreagă a rădăcinii pătrate a valorii **info** din nodul după care are loc inserarea.

a) Definiţi tipurile de date necesare lucrului cu această listă.

b) Scrieţi definiţia completă a subprogramului **Insereaza**.

c) Scrieţi secvenţa de program care, pentru o astfel de listă deja construită, cu adresa primului element memorată în variabila **prim**, inserează după fiecare nod ce reţine în câmpul **info** un număr pătrat perfect, rădăcina pătrată a acelui număr, folosind apeluri ale subprogramului **Insereaza**.

De exemplu dacă iniţial lista conţinea valorile **2→16→36→12→81**, după executarea programului, conţinutul listei este **2→16→4→36→6→12→81→9**.

15. Subprogramul **nrprim** primeşte prin parametrul **n** un număr întreg cu cel mult **4** cifre şi returnează valoarea **TRUE(1)** dacă **n** este număr prim, respectiv **FALSE(0)** în caz contrar. Subprogramul **primulnrprim** primeşte printr-un parametru **p** adresa unui element oarecare al unei liste liniare simplu înlănţuite, alocată dinamic, în care fiecare element reţine în câmpul **număr** un număr întreg cu cel mult **4** cifre şi în câmpul **urm** adresa elementului următor. Subprogramul returnează adresa primului număr prim găsit, parcurgând lista de la nodul **p** până la sfârşit, sau valoarea **NIL/NULL**, dacă nu este găsit nici un număr prim până la sfârşitul listei. Subprogramul **primulnrprim** va utiliza apeluri ale subprogramului **nrprim.**

a) Scrieţi numai antetul subprogramului **nrprim**.

b) Scrieţi definiţia tipurilor de date necesare şi definiţia completă a subprogramului **primulnrprim.**

16. Fie o listă simplu înlănţuită circulară, în care fiecare nod reţine în câmpul **inf** un caracter şi în câmpul **next** adresa nodului următor. Se consideră subprogramele:

- **creare** care primeşte prin parametrul **cuv** un cuvânt de cel mult **20** de litere mici ale alfabetului englez, creează o listă circulară cu toate literele cuvântului **cuv**, în ordinea în care sunt aşezate în cuvânt şi returnează adresa nodului din listă care reţine prima literă din cuvânt.
- **afişare** care afişează pe ecran caracterele din toate nodurile listei circulare pornind de la adresa nodului transmisă prin parametrul **p** şi vizitând toate nodurile listei o singură dată.

O variabilă ce reţine adresa unui nod este de tipul **adnod**.(Un nod al listei este de tipul **nod** -*pentru C/C++*)

. a) Scrieţi antetele pentru subprogramele **creare** şi **afişare**.

b) Definiţi tipurile de date necesare şi scrieţi programul **Pascal** care citeşte de la tastatură un cuvânt de maxim **20** de litere, creează o listă circulară cu toate literele cuvântului şi afişează, câte una pe linie, toate permutările circulare ale cuvântului, utilizând apeluri ale subprogramelor de la punctul a).

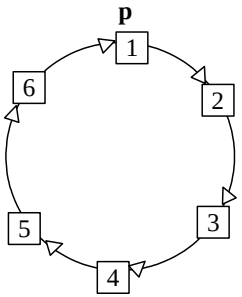
De exemplu, dacă se citeşte de la tastatură cuvântul **carte**, se vor afişa cuvintele **carte**, **artec**, **rteca**, **tecar**, **ecart**.

17. Se consideră o listă circulară simplu înlănţuită, fiecare nod al listei reţinând în câmpul **leg** adresa elementului următor al listei. Componentele listei memorează în câmpul **info**, în ordine, numere naturale consecutive începând cu numărul **1**.

Scrieţi declarările de date şi un subprogram **elimin** ce realizează ştergerea din listă a nodurilor ce memorează numere pătrate perfecte. Subprogramul **elimin** are un singur parametru **p**, reprezentând adresa nodului ce memorează valoarea **1**.

De exemplu, dacă lista conţine iniţial numerele de la 1 la 6, după apelul subprogramului **elimin(p)** aceasta va conţine, în ordine, valorile **2,3,5,6**.

18. Se consideră o listă circulară simplu înlănţuită cu cel puţin **3** elemente. Fiecare nod memorează în câmpul **info** un număr natural şi în câmpul **next** adresa elementului următor din listă. Scrieţi definiţia completă a unui subprogram **verif** care primeşte prin intermediul parametrului **p** adresa unui element oarecare din listă şi returnează valoarea **1** în cazul în care informaţia memorată de nodul de la adresa **p** este media aritmetică a informaţiilor memorate în nodul precedent şi în nodul următor şi returnează valoarea 0 în caz contrar. De exemplu pentru lista din figura alăturată funcţia **verif** va returna valoarea 0.



19. O listă liniară dublu înlănţuită, reţine în fiecare nod, o valoare întregă în cîmpul **info** şi adresele nodului precedent, respectiv următor, în câmpurile **prec** şi respectiv **urm**.

a) Scrieţi definiţia tipului de dată corespunzător unui nod al listei.

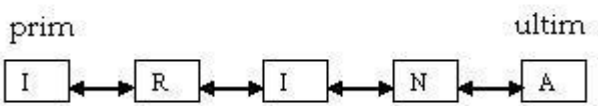
b) Scrieţi definiţia completă a subprogramului cu numele **Sterge** cu trei parametri: **prim**, **ultim** şi **p** care reprezintă adresele primului, ultimului şi respectiv unui nod oarecare al listei şi care şterge din listă nodul de la adresa **p**.

20. Se consideră **x**, un cuvânt care are maxim **30** de caractere, litere ale alfabetului englez. Subprogramul **creare**, creează o listă dublu înlănţuită alocată dinamic care conţine în ordine caracterele cuvântului **x**, câte o literă în fiecare nod al listei.

Subprogramul va avea ca parametri: **x** cuvântul dat, **prim** adresa de început a listei formate şi **ultim,** adresa ultimului nod din listă.

a) Scrieţi definiţia completă a subprogramului **creare**.

De exemplu: pentru cuvântul citit de la tastatură: **IRINA**, lista creată va fi:



b) Scrieţi programul care citeşte de la tastatură un cuvânt, creează folosind apeluri ale subprogramului **creare,** lista dublu înlănţuită ce va conţine literele cuvântului citit şi afişează pe acelaşi rând de ecran, una lângă alta, caracterele din listă, astfel încât să se obţină prenumele persoanei citit invers.

Exemplu: pentru lista creată mai sus se va afişa **ANIRI**.

7. Grafuri

7.1. Grafuri neorientate - Teste grilă

1. Care este numărul **minim** de noduri pe care îl poate conţine un graf neorientat cu 50 de muchii, şi în care 15 noduri sunt izolate?

a. 25 **b.** 66 **c.** 65 **d.** 26

2. Se consideră un graf neorientat cu nodurile: 1,2,3,4,5,6,7,8 şi muchiile: [1,3], [1,7], [2,6], [3,7], [5,2], [5,6], [8,4]. Câte componente conexe are graful?

a. 2 **b.** 3 **c.** 8 **d.** 1

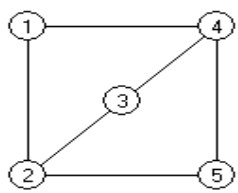
3. Se consideră un graf neorientat cu nodurile: 1,2,3,4,5,6,7,8 şi muchiile: [1,3], [1,7], [2,6], [3,7], [5,2], [5,6], [8,4]. Care este numărul minim de muchii ce pot fi adăugate astfel încât graful să devină conex?

a. 0 **b.** 2 **c.** 3 **d.** 4

4. Care este numărul maxim de vârfuri izolate pe care le poate avea un graf neorientat cu 8 noduri şi 12 muchii?

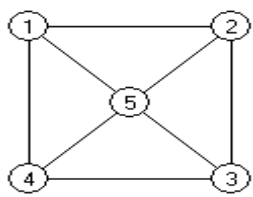
a. 0 **b.** 2 **c.** 3 **d.** 1

5. Se consideră graful neorientat din figura alăturată.
Numărul maxim de muchii ce pot fi eliminate din graf astfel încât în graful parţial rezultat să fie conex este:



a. 0 **b.** 1 **c.** 2 **d.** 3

6. Se consideră graful neorientat din figura alăturată.
Numărul maxim de muchii ce pot fi eliminate din graf astfel încât graful parţial rezultat să fie conex este:

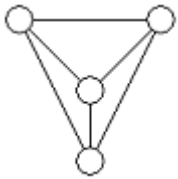


a. 4 **b.** 5 **c.** 3 **d.** 2

7. Într-un graf neorientat G, notăm cu n numărul de vârfuri şi cu m numărul de muchii. Dacă graful este un arbore atunci între n şi m există următoarea relaţie matematică:

a. m=n+2 **b.** n=m-1 **c.** n=m+1 **d.** n=m+2

8. Care este numărul **minim** de muchii care trebuie eliminate astfel încât graful neorientat din figura alăturată să aibă două componente conexe?



a. 5 **b.** 2 **c.** 3 **d.** 4

9. Se consideră graful neorientat cu **6** noduri şi **9** muchii dat prin listele de adiacenţă alăturate. Care este numărul maxim de muchii care se pot elimina astfel încât graful să rămână conex?

```
1: 2 5 6
2: 1 3 4
3: 2 4 6
4: 2 3 5
5: 1 4 6
6: 1 3 5
```

a. 3 **b.** 6 **c.** 5 **d.** 4

10. Dacă **G** este un graf neorientat cu **n** vârfuri şi **n-2** muchii, atunci graful :

a. este conex
b. este arbore
c. este acicilic dacă şi numai dacă are 2 componente conexe
d. nu poate avea vârfuri izolate

11. Fie graful neorientat **G(X,V),** cu **X={1,2,3,4,5}** şi **V={[1,2],[2,3],[3,1],[3,4],[4,5],[5,1],[5,3]}.** Stabiliţi care dintre propoziţiile următoare este adevărată:

a. Numărul vârfurilor de grad par este egal cu numărul vârfurilor de grad impar.
b. Matricea de adiacenţă asociată grafului **G** nu este simetrică faţă de diagonala secundară.
c. Cel mai scurt lanţ de la vârful **1** la vârful **4** are lungimea **3**
d. Subgraful generat de vârfurile **{1,2,4}** nu este conex.

12. Determinaţi câte componente conexe are graful neorientat, a cărui matrice de adiacenţă este dată alăturat:

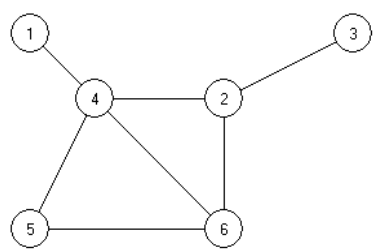
```
0 1 0 0 0 1 1
1 0 0 0 0 1 1
0 0 0 1 0 0 0
0 0 1 0 0 0 0
0 0 0 0 0 0 0
1 1 0 0 0 0 1
1 1 0 0 0 1 0
```

a. 1 **b.** 4 **c.** 3 **d.** 2

13. Numărul maxim de componente conexe ale unui graf neorientat cu **5** noduri şi **4** muchii este:

a. 4 **b.** 2 **c.** 3 **d.** 1

14. Liniile şi coloanele matricei de adiacenţă asociată grafului alăturat sunt numerotate cu 1, 2, …, 6, corespunzător nodurilor grafului. Care dintre următoarele variante este una din liniile matricei de adiacenţă?



a. 0 0 1 1 0 1 **b.** 0 0 0 0 1 0
c. 0 1 1 1 0 0 **d.** 1 1 1 0 1 1

15. Pentru un graf neorientat cu **15** noduri şi **14** muchii, numărul maxim de noduri terminale este:

a. 14 **b.** 7 **c.** 2 **d.** 10

16. Pentru graful neorientat conex cu **7** noduri, în care toate nodurile au acelaşi grad, care dintre următoarele variante **nu** poate fi gradul unui nod?

a. 3 **b.** 2 **c.** 4 **d.** 6

17. Se consideră graful neorientat cu **13** noduri şi mulţimea muchiilor **{[1,4],[2,5],[3,8],[4,7],[4,9],[4,11],[6,3],[6,10],[6,12],[8,6],[13,2]}**. Identificaţi care sunt nodurile care formează componenta conexă cu număr maxim de noduri terminale:

a. 3,6,8,10,12 **b.** 2,5,3,6,8,10,12
c. 1,4,7,9,11 **d.** 2,5

18. Se consideră graful neorientat **G=(X,U)** unde **X={1,2,3,4,5,6}** şi **U={(1,2),(1,3),(6,5),(3,4),(4,5),(4,6)}**.
Stabiliţi care este numărul maxim de muchii care pot fi eliminate pentru a se obţine un graf parţial care să fie conex a lui **G**.

a. 3 **b.** 0 **c.** 2 **d.** 1

19. Identificaţi care din secvenţele următoare reprezintă şirul gradelor nodurilor unui graf complet.

a. 1 2 3 4 **b.** 1 2 12 12 **c.** 5 5 5 5 5 **d.** 4 4 4 4 4

20. Se consideră un graf neorientat dat prin matricea de adiacenţă alăturată. Câte cicluri elementare distincte şi de lungime **3** există în graful din enunţ? (Două cicluri elementare sunt distincte dacă diferă prin cel puţin o muchie).

a. 4 **b.** 0 **c.** 2 **d.** 3

```
0 0 1 0 0 0 0 0
0 0 0 1 1 1 1 1
1 0 0 0 0 0 0 0
0 1 0 0 0 0 0 1
0 1 0 0 0 0 1 0
0 1 0 0 0 0 0 1
0 1 0 0 1 0 0 0
0 1 0 1 0 1 0 0
```

21. Se consideră un graf neorientat cu nodurile: **1,2,3,4,5,6,7,8** şi muchiile **[1,2], [1,5], [2,8], [3,7], [4,5], [5,7], [6,4], [7,6], [8,3], [8,7]**. Care este numărul minim de muchii ce pot fi eliminate astfel încât graful obţinut să aibă trei componente conexe?

a. 3 **b.** 4 **c.** 2 **d.** 5

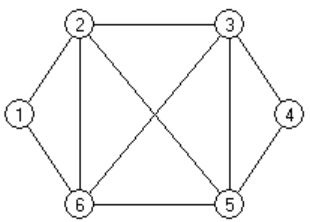
22. Un graf neorientat şi conex are **n** noduri şi **n-1** muchii. Care este numărul minim de muchii ce trebuie adăugate astfel încât să se obţină un ciclu?

a. $\frac{n^2-3\cdot n-2}{2}$ **b.** $\frac{n\cdot(n-1)}{2}$ **c.** 0 **d.** 1

23. Pentru graful neorientat **G=(X,U)** unde **X={1,2,3,4,5,6,7}** şi **U={(1,2),(2,3),(2,7),(1,7),(7,4),(3,4),(4,5),(7,6), (6,5)}** care este numărul minim de muchii care se elimină pentru a obţine un graf cu trei componente conexe?

a. 1 **b.** 3 **c.** 2 **d.** 4

24. Se consideră graful neorientat din figura alăturată. Care dintre succesiunile următoare de noduri reprezintă un lanţ elementar de la nodul **1** la nodul **5**?



a. 1, 6, 2, 3, 6, 5 **c.** 1, 3, 6, 5

b. 1, 2, 6, 3, 5 **d.** 1, 5

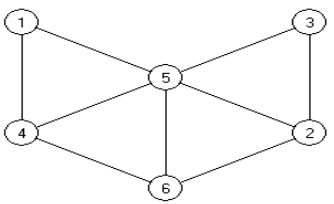
25. Se consideră graful neorientat dat prin lista de muchii: **(1,2), (1,3), (3,4), (3,5), (3,6), (4,8), (4,7)**. Care este numărul minim de muchii ce trebuie eliminate din graf astfel încât acesta să nu mai fie conex?

a. 3 **b.** nicio muchie **c.** 2 **d.** 1

26. Un graf neorientat cu **9** noduri are **2** componente conexe. Ştiind ca în graf nu există noduri izolate, care este numărul maxim de muchii din graf?

a. 22 **b.** 29 **c.** 18 **d.** 16

27. Pentru graful neorientat reprezentat în figura alăturată determinaţi numărul minim de muchii care pot fi eliminate astfel încât graful rămas să nu conţină noduri izolate şi să fie neconex.



a. 4 **b.** 5 **c.** 2 **d.** 3

28. Fie un graf neorientat cu **n=30** noduri şi **m=15** muchii. Numărul componentelor conexe pe care le poate avea acest graf este:

a. cel puţin **1** şi cel mult **30** **b.** cel puţin **10** şi cel mult **15**

c. exact **15** **d.** cel puţin **15** şi cel mult **25**

29. Graful neorientat este dat prin matricea de adiacenţă alăturată. Stabiliţi care dintre următoarele afirmaţii este adevărată:

```
0 1 1 0 0
1 0 1 0 0
1 1 0 0 0
0 0 0 0 1
0 0 0 1 0
```

a. nodurile **2**, **3**, **4** formează un ciclu hamiltonian

b. nodul **5** are gradul **0**

c. nodul **1** este legat printr-un lanţ de nodul **4**

d. nodurile **4** şi **5** aparţin aceleiaşi componente conexe

30. Un graf neorientat cu **n** vârfuri care are proprietatea că oricare două noduri diferite sunt adiacente are un număr de muchii egal cu:

a. **n*(n-1)/2** **b.** **n*n/2**

c. **n*(n+1)/2** **d.** **n*n**

31. Într-un graf neorientat cu **6** noduri oricare două noduri **x, y** sunt adiacente dacă şi numai dacă

x mod 2=y mod 2 | **x%2==y%2**

Care este numărul de componente conexe din graf?

a. **1** **b.** **6** **c.** **3** **d.** **2**

32. Matricea de adiacenţă alăturată corespunde unui graf neorientat care **NU** este de tip:

```
0 1 0 0 1
1 0 1 1 0
0 1 0 1 1
0 1 1 0 1
1 0 1 1 0
```

a. ciclic **b.** hamiltonian **c.** eulerian **d.** conex

33. Se consideră graful neorientat **G = (X, U)** unde **X = {1, 2, 3, 4, 5, 6}** şi **U = {(3,4), (4,6), (3,5), (1,2), (1,3), (6,5), (2,3), (2,5), (1,4)}**. Identificaţi care este numărul minim de noduri care trebuie eliminate pentru a se obţine un subgraf eulerian al lui **G**.

a. **0** **b.** **2** **c.** **1** **d.** **3**

34. Dacă un graf neorientat are **n** noduri şi **p** componente conexe atunci numărul minim de muchii care trebuie adăugate astfel încât graful să devină conex este:

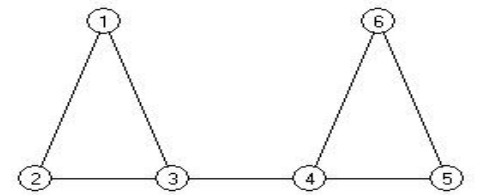
a. **p** **b.** **p-1** **c.** **n-1** **d.** **n**

35. Se consideră un graf neorientat cu 9 noduri şi muchiile [1,2], [4,8], [5,9], [2,3],[7,8], [3,7], [6,9], [6,7], [4,6], [4,5], [1,7]. Numărul minim de muchii care trebuie adăugate pentru ca graful să devină eulerian este:

a. 5 **b.** 0 **c.** 25 **d.** 2

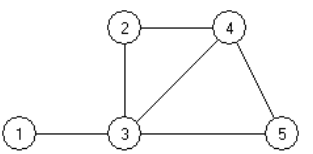
36. Se consideră graful neorientat din figura alăturată:

Care este numărul cel mai mic de muchii care trebuie adăugate pentru ca graful să devină eulerian ?



a. 3 **b.** 2 **c.** 4 **d.** 1

37. Precizaţi care este numărul minim de muchii care trebuie adăugate grafului din figura alăturată, astfel încât acesta să devină eulerian.

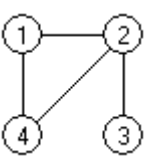


a. 0 **b.** 4 **c.** 2 **d.** 1

38. Se consideră graful neorientat cu 7 noduri şi muchiile: [1,2], [1,4], [1,5], [1,7], [2,3],,7], [3,4], [3,5], [3,7], [4,5], [5,6], [6,7].Care este numărul minim de muchii ce trebuie înlăturate din graf astfel încât să devină eulerian?

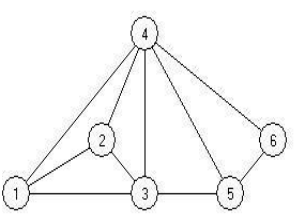
a. 3 **b.** 2 **c.** 1 **d.** 4

39. Se consideră graful neorientat din figura alăturată. Câte grafuri parţiale distincte, diferite de el însuşi, fără vârfuri izolate, se pot obţine?
Două grafuri sunt distincte dacă matricele lor de adiacenţă sunt diferite.



a. 3 **b.** 13 **c.** 5 **d.** 4

40. Specificaţi care este numărul maxim de muchii care pot fi eliminate din graful alăturat, astfel încât acesta să-şi menţină proprietatea de graf hamiltonian



a. 4 **b.** 2 **c.** 1 **d.** 3

41. Dintr-un graf neorientat cu **6** noduri şi **5** muchii, se obţine un graf parţial prin suprimarea a două muchii. Matricea de adiacenţă asociată grafului parţial astfel obţinut, va avea:

a. **6** linii şi **3**coloane
b. **4** linii şi **4** coloane
c. **6** linii şi **4** coloane
d. **6** linii şi **6** coloane

42. Un graf neorientat este reprezentat cu ajutorul listelor de adiacenţă alăturate. Acest graf are:

```
1:(3,5);   5:(3);
2:(4);     6:(7);
3:(1,5);   7:(6);
4:(2);     8:
```

a. **2** componente conexe şi un nod izolat
b. **1** componentă conexă
c. **4** componente conexe
d. **3** componente conexe

43. Fie G un graf neorientat conex cu **20** de vârfuri.Care este numărul minim de muchii ale grafului G?

a. **20**
b. **10**
c. **19**
d. **190**

44. Graful neorientat cu **8** noduri numerotate de la **1** la **8**, este reprezentat cu ajutorul matricei de adiacenţă alăturate. Numărul minim de muchii ce trebuie adăugate pentru ca nodul **2** să fie legat prin lanţuri elementare de lungime **3** de toate nodurile grafului, este:

```
0 1 1 1 0 0 0 0
1 0 1 0 1 1 0 0
1 1 0 0 1 0 0 0
1 0 0 0 0 1 0 0
0 1 1 0 0 1 0 0
0 1 0 1 1 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
```

a. **4**
b. **5**
c. **2**
d. **3**

45. Se dă un graf neorientat cu **75** de noduri numerotate de la **1** la **75**, şi muchiile **[21,40], [30,38], [21,30], [60,75].** Atunci numărul de componente conexe ale grafului este:

a. **69**
b. **71**
c. **2**
d. **73**

46. Câte grafuri neorientate distincte cu **trei** noduri numerotate de la **1** la **3** au muchie între nodul **1** şi nodul **2** ? Două grafuri se consideră distincte dacă matricele lor de adiacenţă sunt diferite.

a. **2**
b. **4**
c. **5**
d. **8**

47. Câte grafuri neorientate distincte cu **n** noduri numerotate **1,2...n** au muchie între nodul **1** şi nodul **2**? Două grafuri se consideră distincte dacă matricele lor de adiacenţă sunt diferite.

a. $2^{n(n-1)/2 -1}$
b. $2^{n(n+1)/2}$
c. $2^{n(n-1)/2}$
d. $2^{n(n-1)/2} -1$

48. Numim graf complementar al unui graf neorientat **G** graful neorientat G_1 cu aceaşi mulţime a nodurilor ca şi **G** şi cu proprietatea că două noduri sunt adiacente în G_1 dacă şi numai dacă nu sunt adiacente în **G.** Dacă **G** are **n** noduri şi **m** muchii, câte muchii are G_1?

a. exact **n(n-1)/2 -m** **b.** minimum **n(n-1)/2 -m**

c. maximum **n(n-1)/2 -m** **d.** exact **n-m**

49. Numărul maxim de muchii dintr-un graf neorientat cu **6** noduri şi **4** componente conexe este:

a. **4** **b.** **1** **c.** **3** **d.** **2**

50. Care este numărul grafurilor parţiale ale unui graf neorientat cu **n** vârfuri şi **m** muchii **?**

a. **n!** **b.** 2^n **c.** **m!** **d.** 2^m

51. Se consideră un graf neorientat cu **7** vârfuri astfel încât între oricare două vârfuri distincte există muchie. Câte lanţuri elementare distincte, care au lungimea 3, extremitatea iniţială vârful 1 şi extremitatea finală vârful 7, există?

a. **10** **b.** **42** **c.** **21** **d.** **20**

52. Se consideră un graf neorientat cu **10** vârfuri şi **37** de muchii.Care dintre următoarele afirmaţii este adevarată?

a. Graful este complet.

b. Suma elementelor matricei de adiacenţă asociată grafului este egală cu 37.

c. Toate vârfurile grafului au gradul 1.

d. Graful nu are vârfuri izolate.

53. Se consideră un graf neorientat cu **10** vârfuri cu proprietatea că există muchie de la vârful **i** la vârful **j** dacă şi numai dacă **i** şi **j** sunt numere prime (numărul 1 se consideră că nu este prim). Care este numărul muchiilor din acest graf?

a. **7** **b.** **6** **c.** **9** **d.** **12**

54. Care dintre următoarele grafuri este un graf eulerian, dar **nu** este hamiltonian? Grafurile sunt precizate prin numărul **n** de noduri şi mulţimea **U** a muchiilor.

a. **n=3, U={[1,2],[1,3],[2,3]}**

b. **n=4, U={[1,2],[1,3],[1,4],[2,3],[2,4],[3,4]}**

c. **n=5, U={[1,3],[1,4],[3,4],[2,4],[4,5],[2,5]}**

d. niciunul dintre grafurile anterioare.

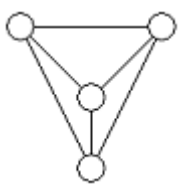
55. Care este numărul maxim de componente conexe pe care le poate avea un graf neorientat cu **6** noduri şi 5 muchii?

a. 4 **b.** 2 **c.** 1 **d.** 3

56. Fie graful neorientat cu **5** noduri şi cu următoarele muchii: **[1,2]**, **[1,3]**, **[3,4]**, **[3,5]**, **[4,5]**. Care este numărul minim de muchii ce trebuie adăugate grafului astfel încât, în graful obţinut toate nodurile să aibă acelaşi grad?

a. 4 **b.** 5 **c.** 6 **d.** 3

57. Care este numărul **maxim** de muchii care pot fi eliminate astfel încât graful parţial obţinut să nu conţină noduri izolate?



a. 4 **b.** 5

c. 2 **d.** 3

58. Fie graful neorientat G cu **n** vârfuri etichetate cu numere de la **1** la **n** şi având proprietatea că între oricare două vârfuri distincte **i** şi **j**, **(1≤i≤n, 1≤j≤n)**, există muchie dacă şi numai dacă **i+j=n**. Precizaţi numărul componentelor conexe ale grafului G.
S-a folosit notaţia **[x]** pentru partea întreagă a numărului **x**.

a. n*(n-1)/2 **b.** [(n+1)/2] **c.** n-1 **d.** [n/2]+1

59. Graful neorientat G cu **n** vârfuri şi **m** muchii are vârfurile etichetate cu $x_1, x_2, x_3, \ldots, x_n$. Care dintre următoarele afirmaţii este corectă, dacă s-a notat cu $d(x_i)$ gradul vârfului x_i?

a. $d(x_1)+d(x_2)+d(x_3)+\ldots+d(x_n)=m-n$

b. $d(x_1)+d(x_2)+d(x_3)+\ldots+d(x_n)=m-1$

c. $d(x_1)+d(x_2)+d(x_3)+\ldots+d(x_n)>n*(n-1)$

d. $d(x_1)+d(x_2)+d(x_3)+\ldots+d(x_n)$ este un număr par

60. Fie un graf neorientat cu **n** vârfuri (**n>1**). Câte valori **1** apar în matricea de adiacenţă a grafului dacă există muchie între oricare două vârfuri distincte?

a. n*(n-1)/2 **b.** n^2 **c.** 0 **d.** n*(n-1)

61. Un graf neorientat cu **n** noduri, cu **n** număr impar mai mare decât **2**, în care fiecare nod are gradul **n-1**, este întotdeauna:

a. graf aciclic (graf care nu conţine nici un ciclu) **b.** arbore
c. graf neconex **d.** graf eulerian

62. Se consideră graful neorientat reprezentat prin matricea de adiacenţă alăturată; atunci graful este

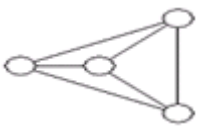
```
0 1 1 1 0
1 0 1 0 1
1 1 0 0 0
1 0 0 0 1
0 1 0 1 0
```

a. eulerian **b.** aciclic (nu conţine niciun ciclu)
c. arbore **d.** hamiltonian

63. Se consideră graful neorientat: **G=(X,U) cu X={1,2,3,4,5,6,7} şi U={[1,3], [2,3], [3,4], [3,5], [5,4], [1,2], [2,5], [2,4], [6,7], [3,6]}.** Care dintre următoarele succesiuni de noduri reprezintă un lanţ hamiltonian în graful dat?

a. (7, 6, 3, 5, 4, 2, 1) **b.** (1, 2, 3, 4, 5, 6, 7)
c. (1, 3, 5, 4, 2, 3, 6) **d.** (4, 5, 3, 6, 7)

64. Care este numărul **minim** de muchii care trebuie eliminate astfel încât graful alăturat să devină eulerian?



a. 2 **b.** 3 **c.** 1 **d.** 0

65. Un graf neorientat este eulerian dacă:

a. este conex şi conţine cel puţin un ciclu elementar
b. conţine un singur ciclu elementar
c. este conex şi suma elementelor de pe fiecare coloană a matricei de adiacenţă este număr par
d. conţine cel puţin un ciclu hamiltonian

66. Se consideră graful neorientat dat prin matricea de adiacenţă alăturată. Care este numărul maxim de noduri ale unui subgraf eulerian al grafului dat?

```
0 1 1 0 0 0 1
1 0 1 1 0 0 1
1 1 0 0 0 1 0
0 1 0 0 1 0 1
0 0 0 1 0 1 0
0 0 1 0 1 0 0
1 1 0 1 0 0 0
```

a. 6 **b.** 3 **c.** 5 **d.** 4

67. Care este numărul minim de muchii care pot fi eliminate din graful neorientat, dat prin listele de adiacenţă alăturate, astfel încât graful să devină eulerian?

```
1:(2,3,5)
2:(1,4)
3:(1,4,5)
4:(2,3,5)
5:(1,3,4)
```

a. 1 **b.** 2 **c.** 3 **d.** 0

68. Considerând un graf neorientat G cu 5 noduri şi matricea de adiacenţă dată alăturat, stabiliţi care dintre următoarele afirmaţii **nu** este adevărată:

```
0 1 1 1 1
1 0 0 0 1
1 0 0 1 0
1 0 1 0 0
1 1 0 0 0
```

a. G este eulerian
b. G este conex
c. G nu este hamiltonian
d. G este aciclic

69. Considerând un graf neorientat G cu 5 noduri, dat prin matricea de adiacenţă alăturată, stabiliţi care dintre următoarele afirmaţii este adevărată:

```
0 1 1 1 1
1 0 1 0 0
1 1 0 0 0
1 0 0 0 1
1 0 0 1 0
```

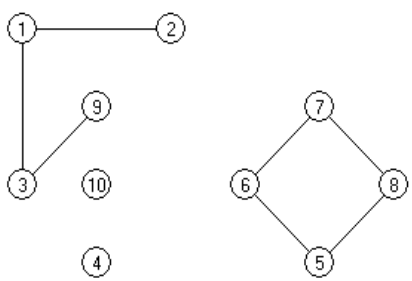
a. G nu este conex
b. G este eulerian
c. G este aciclic
d. G este hamiltonian

70. Considerând un graf neorientat G cu 5 noduri dat prin matricea de adiacenţă alăturată, stabiliţi care dintre următoarele afirmaţii este adevărată:

```
0 1 1 0 1
1 0 1 0 0
1 1 0 0 0
0 0 0 0 1
1 0 0 1 0
```

a. G este aciclic
b. G este conex
c. G este eulerian
d. G este hamiltonian

71. Numărul minim de muchii care trebuie adăugate grafului din desenul alăturat pentru a deveni eulerian este:



a. 5
b. 2
c. 4
d. 3

7.2. Grafuri orientate - Teste grilă

1. Se consideră graful orientat cu 5 noduri numerotate de la 1 la 5 şi cu arcele: **(1,2),(2,1),(2,5),(3,2),(4,3),(5,1),(5,2),(5,4)**.

Determinaţi gradul intern al nodului cu gradul extern maxim.

a. 3 **b.** 1 **c.** 2 **d.** 0

2. Suma gradelor interne ale tuturor vârfurilor unui graf orientat este totdeauna egală cu:

a. numărul valorilor de **1** aflate sub diagonala principală în matricea de adiacenţă

b. suma tuturor valorilor aflate deasupra diagonalei principale în matricea de adiacenţă

c. produsul gradelor externe ale tuturor vârfurilor grafului

d. suma gradelor externe ale tuturor vârfurilor grafului

3. Un graf orientat este reprezentat prin matricea de adiacenţă dată alăturat. Precizaţi care sunt nodurile pentru care gradul interior este mai mare decât gradul exterior.

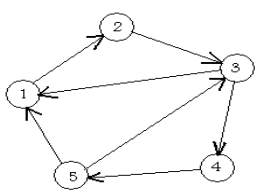
```
0 1 1 0 0 0
0 0 1 1 0 1
1 1 0 1 0 0
0 0 0 0 1 0
0 1 0 0 0 0
0 1 0 0 1 0
```

a. 2, 4, 5 **b.** 2, 4, 5, 6 **c.** 1, 4, 5 **d.** 1, 3, 6

4. Numărul de grafuri orientate cu **n** vârfuri este:

a. 2^n **b.** $2^{n(n-1)}$ **c.** $2^{\frac{n(n-1)}{2}}$ **d.** **2n**

5. Fie graful orientat reprezentat în figura alăturată. Câte dintre vârfurile grafului au gradul intern egal cu **2**?



a. 3 **b.** 1

c. 0 **d.** 2

6. Gradul intern pentru nodul cu eticheta **i** dintr-un graf orientat la care se cunoaşte matricea de adiacenţă este egal cu numărul de cifre egale cu **1** aflate pe:

a. linia **i** **c.** diagonala secundară

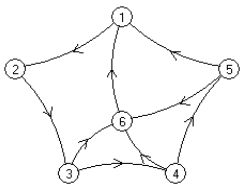
b. diagonala principală **d.** **coloana i**

7. Fie **G** un graf orientat cu **6** vârfuri dat prin matricea de adiacenţă alăturată. Precizaţi câte dintre vârfurile grafului au gradul intern egal cu gradul extern?

```
0 0 0 0 1 0
1 0 0 1 0 1
0 0 0 0 0 0
1 0 0 0 1 0
0 1 0 0 0 0
0 0 0 0 1 0
```

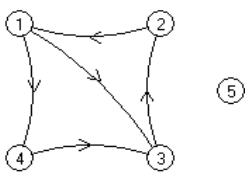
a. 2 **b.** 1 **c.** 4 **d.** 3

8. Considerând graful orientat din figura alăturată, stabiliţi câte dintre vârfurile grafului au gradul extern (exterior) egal cu dublul gradului intern (interior).



a. 2 **b.** 1 **c.** 0 **d.** 3

9. Considerând graful orientat din figura alăturată, stabiliţi câte dintre vârfurile grafului au gradul extern (exterior) egal cu gradul intern (interior).



a. 2 **b.** 3 **c.** 1 **d.** 0

10. Într-un graf orientat cu **10** vârfuri numerotate de la **1** la **10** există arce numai între perechile de vârfurile **i** şi **j, i≠j** cu proprietatea că **i** este divizor al lui **j** (**i** fiind extremitatea iniţială şi **j** extremitatea finală a arcului). Numărul de valori egale cu **1** din matricea de adiacenţă corespunzătoare grafului este:

a. 17 **b.** 10 **c.** 30 **d.** 34

11. Graful orientat **G=(X,U)** are **20** de vârfuri numerotate de la **1** la **20** şi arce între vârfurile numerotate **i** şi **j** care îndeplinesc condiţiile: **i** este număr de o singură cifră iar **j** este un număr de două cifre ce are în scrierea sa cifra **i**. Numărul valorilor de **1** din matricea de adiacenţă asociată grafului **G** este:

a. 20 **b.** 19 **c.** 10 **d.** 15

12. Care e numărul minim de arce pe care trebuie să le conţină un graf cu **5** vârfuri care astfel încât oricum ar fi acestea plasate să existe cel puţin un drum între oricare două vârfuri.

a. 10 **b.** 9 **c.** 20 **d.** 17

13. Se consideră graful orientat dat prin matricea de adiacenţă alăturată. Care este lungimea maximă a unui drum elementar de la vârful **1** până la vârful **5**?

```
0 1 1 0 0 0
0 0 0 1 1 1
0 0 0 0 0 0
0 0 1 0 0 1
0 0 1 0 0 0
0 1 0 0 1 0
```

a. 4 **b.** 3 **c.** 1 **d.** 5

14. Care dintre următoarele arce trebuie adăugat unui graf orientat cu **5** noduri şi cu matricea de adiacenţă alăturată astfel încât în acest graf să existe cel puţin un drum între oricare două vârfuri?

```
0 1 0 1 0
0 0 1 0 0
0 0 0 0 0
0 0 0 0 1
1 0 0 0 0
```

a. (3 , 5) **b.** (4 , 1) **c.** (5 , 3) **d.** (3 , 2)

15. Matricea de adiacenţă a unui graf orientat cu 8 noduri şi 16 arce este simetrică faţă de diagonala principală. Care dintre următoarele afirmaţii este adevărată pentru acest graf?

a. Fiecare nod al grafului are gradul interior diferit de gradul exterior
b. **Fiecare nod al grafului are gradul interior egal cu gradul exterior**
c. Numărul de valori egale cu **1** din matricea de adiacenţă este impar
d. Graful nu conţine nici un drum

16. Pentru un graf orientat dat, notăm cu **Se** suma gradelor exterioare ale tuturor nodurilor grafului şi cu **Si** suma gradelor interioare ale tuturor nodurilor grafului. Care dintre următoarele relaţii matematice este adevărată?

a. **Se≠Si** **b.** **Se=Si** **c.** **Se<Si** **d.** **Se>Si**

17. Fie **G=(V,E)** un graf orientat în care mulţimea nodurilor este **V={1,2,…,10}**, iar mulţimea arcelor este **E={(i,j)∈VxV| i≠j** şi **j mod i=0}** (prin **a mod b** am notat restul împărţirii lui **a** la **b**). Stabiliţi care dintre următoarele afirmaţii este adevărată:

a. Pentru oricare pereche de noduri **i** şi **j (i≠j)** există cel puţin un drum de la **i** la **j** şi cel puţin un drum de la **j** la **i**
b. **pentru orice nod al grafului G suma dintre gradul interior şi gradul exterior este nenulă**
c. toate vârfurile grafului **G** au gradul interior egal cu gradul exterior
d. graful **G** conţine circuite

18. Fie graful orientat **G=(V,E)** unde mulţimea nodurilor este **V={1,2,3,4,5,6,7}**, iar mulţimea arcelor este **E={[1,2],[1,6],[2,5], [2,6],[3,4],[4,3],[6,2], [6,5],[3,7],[4,7]}**. Numărul nodurilor grafului **G** care au gradul exterior egal cu **0** este:

a. 1 **b.** 3 **c.** 0 **d.** 2

19. Considerând graful orientat **G** cu **6** noduri reprezentat prin intermediul listelor de adiacenţă alăturate, stabiliţi câte dintre vârfurile sale au gradul intern egal cu gradul extern:

```
1: 5
2: -
3: 2 4
4: 2 3
5: 2 4
6: 1 2 3 4 5
```

a. 4 **b.** 1 **c.** 3 **d.** 2

20. Câte dintre nodurile grafului orientat cu **6** noduri şi cu matricea de adiacenţă alăturată au gradul interior egal cu gradul exterior?

```
0 1 0 0 0 0
1 0 1 0 0 1
0 0 0 1 0 1
0 1 1 0 1 0
0 0 0 1 0 1
1 0 1 1 0 0
```

a. 2 **b.** 1
c. 4 **d.** 3

21. Lungimea unui drum elementar într-un graf orientat cu **n** vârfuri poate fi:

a. ∞ **b.** n+1 **c.** n **d.** n-1

22. Fie graful orientat cu **5** vârfuri reprezentat prin matricea de adiacenţă alăturată.
Care este mărimea celui mai lung drum elementar din graf?

```
0 0 0 1 1
0 0 1 0 1
0 0 0 0 0
0 1 1 0 0
0 0 0 0 0
```

a. 2 **b.** 1 **c.** 3 **d.** 4

23. Fie graful orientat G cu **5** noduri, reprezentat prin matricea de adiacenţă alăturată. Precizaţi lungimea celui mai mare drum elementar din graful G?

```
0 1 1 0 0
0 0 0 1 0
0 1 0 1 0
0 0 0 0 0
1 0 0 1 0
```

a. 5 **b.** 3 **c.** 2 **d.** 4

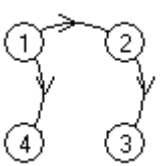
24. Fie graful orientat cu **5** vârfuri şi următoarele arce: **[1,2], [1,4], [3,1], [3,2], [4,5], [4,2], [5,1]**. Câte circuite conţine acest graf?

a. 3 **b.** 4 **c.** 2 **d.** 1

25. Se consideră un graf orientat cu **6** vârfuri şi arcele: **(1,4), (1,5), (2,3), (2,4), (3,4), (4,3), (4,6), (5,4), (6,4)**. Gradul interior al vârfului **4** este:

a. 7 **b.** 3 **c.** 2 **d.** 5

26. Care este numărul **minim** de arce care trebuie adăugate grafului orientat din figura alăturată astfel încât oricare două vârfuri să fie unite prin drumuri elementare?



a. 1 **b.** 3 **c.** 0 **d.** 2

27. Se consideră graful orientat cu **8** noduri, definit cu ajutorul listelor de adiacenţă alăturate. În acest graf, nodul **1** este legat prin drumuri de lungime **2** de nodurile:

```
1: 4, 5, 6
2: 3, 4
3: 4
4: 3, 6
5: 4, 1
6: 1, 4
7: 1, 8
8:
```

a. 7,8 **b.** 5,6,4 **c.** 3,4,6 **d.** 2

28. Un graf orientat, este memorat cu ajutorul listelor alăturate de adiacenţă. Numărul nodurilor care au gradul interior egal cu gradul exterior este:

```
1: 5
2: 4
3: 5
4: 1, 2
5: 2, 3, 4
```

a. 2 **b.** 4 **c.** 1 **d.** 3

29. Se consideră graful orientat dat prin matricea de adiacenţă alăturată, ale cărui noduri sunt numerotate de la **1** la **4** corespunzător liniilor matricei. Să se determine care sunt nodurile care au gradul intern egal cu **2** :

```
0 0 0 1
1 0 0 0
1 1 0 0
0 1 1 0
```

a. nici nodul **1** şi nici nodul **2**
b. atât nodul **1** cât şi nodul **2**
c. numai nodul **2**
d. numai nodul **1**

30. Un graf orientat are cinci noduri numerotate cu **1, 2, 3 ,4, 5** şi patru arce: **[1,2], [2,1], [2,3], [3,4]**. Prin eliminarea nodului **2** şi a arcelor incidente cu acesta obţinem:

a. un subgraf cu patru noduri şi un arc
b. un subgraf cu două noduri şi niciun arc
c. un graf parţial
d. un subgraf cu cinci noduri şi trei arce

31. Care dintre următoarele secvenţe de noduri reprezintă un drum în graful orientat dat prin matricea de adiacenţă alăturată, ştiind că nodurile sunt numerotate de la **1** la **5** corespunzător liniilor şi coloanelor tabloului?

```
0 1 0 0 0
0 0 0 1 0
0 0 0 0 0
0 1 1 0 0
0 0 0 1 0
```

a. 1,5,4,3 **b.** 1,2,4,3 **c.** 5,4,3,1 **d.** 2,4,3,1

32. Într-un graf orientat cu **n** noduri, gradul extern al unui vârf poate fi maximum:

a. n-1 **b.** 1 **c.** n+1 **d.** 2

33. Într-un graf orientat **G(X,V)** cu **6** noduri numerotate cu numere distincte de la **1** la **6**, există arc de la nodul **i** la nodul **j** dacă şi numai dacă **i<j** şi **j-i>1**. Numărul de noduri din graf care au gradul interior mai mare decât gradul exterior este:

a. 3 **b.** 0 **c.** 2 **d.** 1

34. Pentru un graf orientat **G(X,V)** cu **n** noduri numerotate cu numerele distincte **1,2,..,n**, şi reprezentat prin matricea de adiacenţă **a**, secvenţa de instrucţiuni alăturată descrisă în limbajul pseudocod determină în variabila **nr**:

```
nr←0
citeşte k
        {k natural,k≤n}
pentru i←1,n execută
  dacă a[ki]=1 atunci
       nr ←nr+1
```

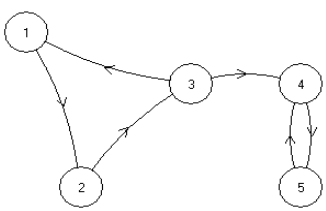
a. gradul nodului **k**
b. gradul exterior al nodului **k**
c. gradul interior al nodului **k**
d. numărul de elemente egale cu **1** din matricea de adiacenţă

35. Graful orientat **G** cu **10** noduri, reprezentat prin listele de adiacenţă alăturate, are:

```
1: 4 6
2: 1
3:
4: 6
5: 7 9
6:
7:
8:
9: 8
10:
```

a. Două drumuri distincte de la nodul **2** la nodul **6**
b. Un drum de la nodul **7** la nodul **8**
c. Un circuit care conţine nodurile **1,4,6**
d. Două drumuri distincte de la nodul **5** la nodul **8**

36. Matricea drumurilor unui graf orientat este o matrice de dimensiune **nxn**, definită astfel: $a_{ij}=1$ dacă există cel puţin un drum de la nodul **i** la nodul **j** şi, respectiv $a_{ij}=0$ dacă nu există niciun drum de la **i** la **j**. Care este matricea drumurilor pentru graful alăturat?



a.
```
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
0 0 0 1 1
0 0 0 1 1
```
b.
```
0 1 1 1 1
1 0 1 1 1
1 1 0 1 1
0 0 0 0 1
0 0 0 1 0
```
c.
```
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
0 0 1 1 1
0 0 0 1 1
```
d.
```
0 1 0 0 0
0 0 1 0 0
1 0 0 1 0
0 0 0 0 1
0 0 0 1 0
```

37. Considerăm un graf orientat cu **n** vârfuri şi **m** arce . Ce valoare se obţine prin însumarea elementelor matricei de adiacenţă asociată grafului?

a. **n** **b.** **2*m** **c.** **m/2** **d.** **m**

38. Se consideră graful orientat cu **5** noduri, numerotate de la **1** la **5**, reprezentat cu ajutorul matricei de adiacenţă alăturată. Ce arc trebuie adăugat astfel încât graful să conţină cel puţin un circuit elementar de lungime **5**?

```
0 1 0 0 0
1 0 0 0 0
0 1 0 1 0
1 0 0 0 0
0 1 1 0 0
```

a. **(5,2)** **b.** **(5,4)** **c.** **(4,5)** **d.** **(2,5)**

39. Se consideră graful orientat dat prin matricea de adiacenţă alăturată.

Stabiliţi care dintre următoarele afirmaţii este adevărată.

```
0 1 1 0
0 0 0 0
0 1 0 1
0 0 0 0
```

a. graful conţine un circuit

b. există noduri cu gradul intern egal cu gradul extern

c. graful conţine un singur vârf cu gradul intern `0`

d. graful nu conţine niciun drum elementar (un drum se numeşte elementar dacă vârfurile din componenţa sa sunt distincte)

40. Considerăm un graf orientat `G` cu `4` noduri şi cu gradele externe ale acestora: `2,1,0,2` Care dintre variantele următoare poate reprezenta şirul gradelor interne ale lui `G`?

a. `1,1,1,1` **b.** `1,1,3,0` **c.** `1,1,2,2` **d.** `1,3,2,0`

41. Se consideră graful orientat `G=(V, E)` unde `V={1,2,3,4,5,6}` şi `E={[1,2],[6,1],[2,5],[2,3],[4,5],[3,4],[6,5]}`. Care este numărul maxim de arce dintr-un drum elementar al grafului (drum cu noduri distincte)?

a. `3` **b.** `6` **c.** `4` **d.** `5`

42. Fie graful orientat cu nodurile numerotate cu numerele distincte `1,2,3,4,5` şi care conţine arcele: `(1,2),(1,4),(1,5),(5,4),(4,3),(3,2),(3,1)`. Care din următoarele succesiuni reprezintă un drum elementar (cu toate nodurile distincte)?

a. `1,2,3` **b.** `1,5,4,3,2` **c.** `3,1,4,3,2` **d.** `1,2,5,4,3`

43. Fie graful orientat `G` cu `n=6` noduri dat prin listele de adiacenţă: `1`: (`2`,`3`,`4`), `2`: (`3`, `5`), `3`: (`2`, `4`), `4`: (`5`), `5`: (`6`), `6`: (`4`). Care este lungimea celui mai scurt drum de la nodul `1` la nodul `6`?

a. `2` **b.** `3` **c.** `1` **d.** `4`

44. Fie graful orientat `G` cu `n=5` noduri, dat prin următoarele liste de adiacenţă: `1`: (`2`, `3`), `2`: (`3`, `4`), `3`: (`4`, `5`), `4:` (`1`, `2`), `5`: (`4`).

Care dintre următoarele propoziţii este **falsă**?

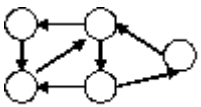
a. există cel puţin un nod în graful `G` care are gradul intern egal cu cel extern

b. există cel puţin un drum între oricare două noduri ale grafului `G`

c. graful `G` nu are circuite

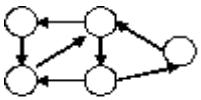
d. graful `G` are `9` arce

45. Care este numărul minim de arce ce trebuie eliminate astfel încât graful din desenul alăturat să nu conţină niciun circuit?



a. 1 **b.** 3 **c.** 0 **d.** 2

46. Care este numărul de circuite elementare distincte în graful din figura din dreapta? (Două circuite elementare sunt distincte dacă diferă prin cel puţin un arc.)



a. 4 **b.** 3 **c.** 0 **d.** 2

47. Se consideră un graf orientat cu **6** noduri numerotate cu **1, 2,...,6** şi cu mulţimea arcelor formată **doar** din arcele:

- de la fiecare nod numerotat cu un număr neprim **i** (**i>1**) la toate nodurile numerotate cu numere ce aparţin mulţimii divizorilor proprii ai lui **i** (divizori diferiţi de **1** şi de **i**);
- de la nodul numerotat cu **1** la nodul numerotat cu **2**;
- de la fiecare nod numerotat cu un număr prim **i** la nodul numerotat cu **i+1**.

Stabiliţi care este numărul de circuite elementare distincte conţinute de graful din enunţ. (Două circuite sunt distincte dacă diferă prin cel puţin un arc).

a. 1 **b.** 2 **c.** 3 **d.** 0

48. Un graf orientat are **8** arce şi fiecare nod al grafului are gradul interior un număr nenul. Doar **două** dintre noduri au gradul interior un număr par, restul nodurilor având gradele interioare numere impare. Care este numărul maxim de noduri pe care poate să le aibă graful?

a. 7 **b.** 8 **c.** 5 **d.** 6

49. Se consideră un graf orientat cu **6** noduri numerotate cu **1, 2,...,6** şi cu mulţimea arcelor formată **doar** din arcele:

- de la fiecare nod numerotat cu număr neprim **i** (**i>1**) la toate nodurile numerotate cu numere ce aparţin mulţimii divizorilor proprii ai lui **i** (divizori diferiţi de **1** şi **i**);
- de la nodul numerotat cu **1** la nodul numerotat cu **2**;
- de la fiecare nod numerotat cu un număr prim **i** la nodul numerotat cu **i+1**.

Stabiliţi câte noduri din graf au suma dintre gradul intern şi cel extern egală cu **3**.

a. 1 **b.** 6 **c.** 2 **d.** 0

50. Un graf orientat are **8** arce şi fiecare nod al grafului are gradul exterior un număr nenul. Doar **două** dintre noduri au gradul exterior un număr impar, restul având gradele exterioare numere pare. Care este numărul maxim de noduri pe care le poate avea graful?

a. 4 **b.** 8 **c.** 3 **d.** 5

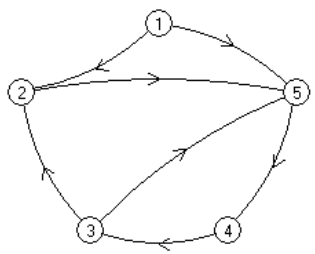
51. Fie graful orientat cu **5** noduri şi arcele **(1,2), (1,5), (2,5), (2,4), (3,2), (4,3), (4,5)**. Care este numărul minim de arce care trebuie adăugate grafului astfel încât să existe cel puţin un drum între oricare două vârfuri?

a. 1 **b.** 0 **c.** 3 **d.** 2

52. Un graf orientat are **11** vârfuri numerotate de la **1** la **11**. Între oricare două vârfuri ale sale, **x** şi **y** (**x** ≠**y**), există atât arcul de la **x** la **y** cât şi arcul de la **y** la **x** dacă şi numai dacă restul împărţirii lui **x** la **3** este egal cu restul împărţirii lui **y** la **3**. Care este numărul minim de arce care trebuie adăugate acestui graf astfel încât să existe cel puţin un drum între oricare două vârfuri ale sale.

a. 6 **b.** 4 **c.** 2 **d.** 3

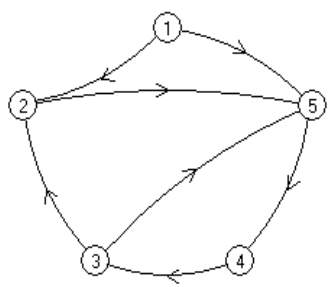
53. Se consideră graful orientat din figura alăturată. Câte circuite elementare disticte are graful?



a. 4 **c.** 1

b. 3 **d.** 2

54. Se consideră graful orientat din figura alăturată. Câte perechi de vârfuri de forma **(x,y)**, cu **x<y**, respectă proprietatea că există cel puţin un drum de la **x** la **y** şi cel puţin un drum de la **y** la **x**?



a. 10 **c.** 6

b. 4 **d.** 8

55. Fie un graf orientat dat care are **5** vârfuri numerotate **1,2,3,4,5** şi arcele: **(2,1), (2,3), (2,4), (3,4), (1,5), (5,4)**. Numărul circuitelor elementare distincte (care diferă prin cel puţin un arc) din graful din enunţ este egal cu:

a. 3 **b.** 0 **c.** 2 **d.** 1

56. Se consideră graful orientat dat prin matricea de adiacenţă alăturată, graf cu 6 noduri numerotate de la 1 la 6 corespunzător liniilor şi coloanelor matricei. Care dintre următoarele este o pereche de noduri `i j` astfel încât există un drum elementar de la `i` către `j`?

```
0 0 1 0 0 1
1 0 0 0 1 0
0 0 0 1 0 0
0 0 1 0 0 0
0 1 0 0 0 0
1 0 0 0 0 0
```

a. `6 5` **b.** `5 4` **c.** `4 6` **d.** `4 5`

57. Se consideră graful orientat `G = (X, U)` unde `X = {1, 2, 3, 4, 5, 6}` şi `U = {(1,2), (1,5), (1,6), (2,3), (3,5), (4,1), (5,4)}`. Identificaţi care sunt nodurile accesibile din toate celelalte noduri ale grafului prin intermediul unor drumuri elementare.

a. `6` **b.** `1 5` **c.** `1 2 3 5` **d.** `4 5`

58. Graful neorientat reprezentat prin listele de adiacenţă alăturate se transformă în graf orientat astfel: fiecare muchie `[i,j]`, cu `i<j`, devine arcul `(i,j)`. În graful orientat astfel obţinut lungimea celui mai scurt drum de la vârful `1` la vârful `5` este:

```
1:(2,3)
2:(1,3,5)
3:(1,2,4)
4:(3,5)
5:(2,4)
```

a. `4` **b.** `1` **c.** `2` **d.** `3`

59. Se consideră un graf orientat dat prin matricea de adiacenţă alăturată. Stabiliţi care este numărul nodurilor din graf care au proprietatea că diferenţa absolută a gradelor (intern si extern) este egală cu `1` ?

```
0 1 0 0 1
0 0 1 1 0
1 0 0 0 0
0 0 1 0 1
0 1 0 0 0
```

a. `4` **b.** `3` **c.** `2` **d.** `5`

60. Fie graful orientat cu `8` vârfuri şi arcele `[1,2], [2,3], [3,1], [4,5], [5,6],[5,7],[6,7],[7,4],[8,7].` Numărul de vârfuri cu proprietatea că gradul interior este egal cu gradul exterior este:

a. `2` **b.** `7` **c.** `0` **d.** `5`

61. Fie graful orientat cu `7` vârfuri, numerotate de la `1` la `7` şi listele de adiacenţă `L1={2,3,4}`, `L2={3,4}`, `L3={4,6}`, `L4={5,6}`, `L5={2,7}`, `L6={4,7}`, `L7={2,4}`. Care este vârful (care sunt vârfurile) cu gradul interior maxim?

a. `3,6,7` **b.** `1` **c.** `2` **d.** `4`

62. Se consideră graful orientat dat prin matricea de adiacenţă alăturată.
Stabiliţi câte dintre nodurile grafului au gradul interior (intern) egal cu gradul exterior (extern).

```
0 1 0 1 0
0 0 0 0 1
0 1 0 0 0
0 1 1 0 0
0 1 0 1 0
```

a. `2` **b.** `1` **c.** `0` **d.** `3`

63. Considerăm un graf orientat cu nodurile numerotate cu numere distincte **1, 2, 3, …** Graful este reprezentat printr-o matrice de adiacenţă **A**.
Precizaţi care este semnificaţia sumei valorilor dintr-o coloană oarecare **x** a matricei **A**.

a. reprezintă numărul arcelor care au ca extremitate finală nodul numerotat cu numărul **x**

b. reprezintă numărul drumurilor care nu trec prin nodul numerotat cu numărul **x**

c. reprezintă numărul drumurilor care trec prin nodul numerotat cu numărul **x**

d. reprezintă numărul arcelor care pleacă din nodul numerotat cu numărul **x**

64. Se consideră graful orientat **G = (X, U)** unde **X = {1, 2, 3, 4, 5, 6}** şi **U = {(1,2), (1,5), (1,6), (2,3), (3,5), (4,1), (5,4)}**.
Identificaţi care sunt nodurile accesibile din toate celelalte noduri ale grafului prin intermediul unor drumuri elementare.

a. 6 **b.** 1 5 **c.** 1 2 3 5 **d.** 4 5

65. Se consideră graful orientat **G=(X,U)** unde **X={1,2,3,4,5,6,7,8,9}** şi **U={(2,1), (1,6), (2,5), (2,3), (3,4), (4,6), (5,7), (4,8), (8,9)}.**
Care sunt nodurile legate de nodul **2** prin drumuri a căror lungime este egală cu cea a drumului de lungime minimă dintre nodurile **2** şi **6** ?

a. 7 4 **b.** 8 2 **c.** 5 8 9 **d.** 1 5 3

66. Precizaţi care dintre nodurile grafului orientat a cărui matrice de adiacenţă este reprezentată alăturat, au gradul interior egal cu gradul exterior.

0 1 0 0 0 0 0 0	
0 0 1 0 0 0 1 0	
0 0 0 1 1 1 0 0	
0 0 0 0 0 0 1 0	
0 0 0 1 0 0 1 0	
0 1 0 1 1 0 0 0	
1 0 1 0 0 1 0 0	
0 0 0 0 0 0 0 0	

a. 1,2,5,7,8 **b.** 1,2,5,6,8

c. 2,5,6,7,8 **d.** 1,2,5,7,6

67. Precizaţi care este lista de adiacenţă corespunzătoare nodului **6**, pentru graful orientat reprezentat prin matricea de adiacenţă alăturată.

0 1 0 0 0 0	
0 0 1 0 0 1	
0 1 0 1 0 1	
0 0 1 0 1 0	
0 0 0 0 0 1	
1 0 1 1 0 0	

a. 1, 3, 4 **b.** 1, 3, 5

c. 2, 3, 5 **d.** 2, 3, 4

68. Se consideră un graf orientat cu **8** noduri, numerotate de la **1** la **8** şi arcele **[1,2]**, **[1,8]**, **[2,3]**, **[2,7]**, **[3,2]**, **[5,8]**, **[6,5]**, **[6,8]**, **[7,3]**, **[7,4]**, **[8,6]**, **[8,7]**. Precizaţi care este nodul la care se poate ajunge, din oricare alt nod al grafului, parcurgând drumuri ale grafului.

a. 3 **b.** 4 **c.** 1 **d.** 2

69. Se consideră graful orientat cu **6** noduri şi arcele **[1,2]**, **[1,6]**, **[2,1]**, **[2,3]**, **[2,4]**, **[2,6]**, **[3,2]**, **[3,4]**, **[3,5]**, **[3,6]**, **[4,3]**, **[4,5]**, **[4,6]**, **[5,4]**, **[6,5]**. Câte drumuri elementare de la nodul **1** la nodul **6** există?

a. 5 **b.** 8 **c.** 7 **d.** 6

70. Se consideră graful orientat cu **6** noduri dat prin matricea de adiacenţă alăturată. Stabiliţi câte perechi neordonate de noduri **(a,b)** există astfel încât **există drum** fie de la **a** către **b**, fie de la **b** către **a**, dar nu amândouă. La numărare ţineţi cont de faptul că, de exemplu, perechea neordonată **(2,4)** este una şi aceeaşi cu perechea **(4,2)**.

```
0 1 0 0 1 0
1 0 0 1 0 0
0 1 0 0 1 1
0 1 0 0 0 0
1 1 0 0 0 0
0 0 1 1 0 0
```

a. 3 **b.** 8 **c.** 4 **d.** 6

71. Se consideră un graf orientat cu **4** noduri etichetate cu numere de la **1** la **4** şi cu arcele **(1,2) (1,3) (2,1) (2,3) (2,4) (4,2) (4,3)**. Care dintre nodurile grafului au gradul interior mai mare decât gradul exterior?

a. 1, 2 şi 4 **b.** 3 **c.** 3 şi 4 **d.** 3 şi 2

7.3. Arbori - Teste grilă

1. Care dintre următoarele matrice este matricea de adiacenţă a unui arbore cu 4 noduri?

a.
```
0 1 0 1
0 0 1 0
1 0 0 0
1 0 1 0
```

b.
```
0 0 1 0
0 0 0 1
1 0 0 0
0 1 0 0
```

c.
```
0 1 1 1
1 0 1 0
1 1 0 0
1 0 0 0
```

d.
```
0 0 1 0
0 0 0 1
1 0 0 1
0 1 1 0
```

2. Se consideră un arbore. Care dintre următoarele afirmaţii este adevărată?

a. are cel puţin un nod izolat
b. toate nodurile au grad par
c. are cel puţin două componente conexe
d. este aciclic

3. Fie graful neorientat dat prin matricea de adiacenţă alăturată. Numărul de muchii ce trebuie eliminate pentru ca graful să devină arbore este:

```
0 1 1 0 0 0 0
1 0 0 1 0 0 0
1 0 0 0 1 0 0
0 1 0 0 0 1 0
0 0 1 0 0 0 1
0 0 0 1 0 0 1
0 0 0 0 1 1 0
```

a. 2 **c.** 0 **d.** 1

b. nu se poate obţine arbore prin eliminări de muchii

4. Numărul de noduri care au gradul **1** într-un graf neorientat conex şi aciclic cu **n** noduri (**n>1**) este:

a. **mai mare sau cel puţin egal cu 2** **b.** exact **n-1** **c.** exact **1** **d.** **0** sau **1**

5. Fie arborele cu **8** noduri şi cu muchiile **[1,2]**, **[1,3]** , **[1,4]** , **[4,5]** , **[6,4]** , **[1,8]** , **[4,7].** Câţi vectori de taţi distincţi se pot construi pentru acest arbore? Doi vectori de taţi sunt distincţi dacă în cei doi vectori există cel puţin o poziţie pentru care elementele din respectivele poziţii sunt distincte.

a. **40320** **b.** **7** **c.** **28** **d.** **8**

6. Se consideră graful neorientat **G=(X,U) X={1,2,3,4,5,6} U={[1,2], [2,3], [2,4], [2,6], [1,5], [5,6]}**. Pentru a transforma graful într-un arbore, putem elimina:

a. **muchiile [1,5] şi [5,6]** **b.** **nodul 3 si muchiile incidente lui**
c. **nodul 4 si muchiile incidente lui** **d.** **muchia [2,6]**

7. Fie **G** un graf neorientat conex cu **100** de noduri şi **2007** muchii. Numărul de muchii care trebuie eliminate din **G** astfel încât acesta să devină arbore este:

a. **1237** **b.** **1907** **c.** **1007** **d.** **1908**

8. Fie **G=(V,E)** un arbore în care **V={1,2,...,n}**.

Ştiind că şi **G'=(V ∪ {n+1},E')** este deasemenea un arbore, stabiliţi care dintre următoarele propoziţii este adevărată (notaţia **|M|** reprezintă numărul elementelor unei mulţimi **M**):

a. **|E'|=|E|** **b.** **|E'|=|E|+1** **c.** **|E'|=|E|-1** **d.** **|E'|=|E|+2**

9. Prin înălţimea unui arbore cu rădăcină înţelegem numărul de muchii ale celui mai lung lanţ elementar care are una dintre extremităţi în rădăcina arborelui. Dacă arborele **T** este dat prin următorul vector de taţi: **4,5,1,0,4,5,6,1,4**, atunci care este înălţimea sa?

a. **1** **b.** **2** **c.** **3** **d.** **4**

10. Dacă **G** este un graf neorientat cu proprietatea că între orice două vârfuri ale sale există un unic lanţ elementar, atunci **G** este:

a. graf eulerian
b. arbore
c. graf hamiltonian
d. un graf cu toate gradele numere impare

11. Un arbore cu **10** noduri are următorul vectorul de taţi: **T=[4,4 2,5,0,5,8,6,8,8].** Câte noduri frunză (terminale) are acest arbore?

a. **5** **b.** **3** **c.** **4** **d.** **6**

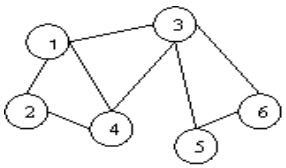
12. Se construieşte un arbore în care nodul rădăcină memorează valoarea **20** iar fiecare nod neterminal are ca descendenţi direcţi noduri în care se păstrează **divizorii proprii** ai valorii din nodul părinte (numărul natural **d** este dizivor propriu al numărului natural **a**, dacă **d** este divizor al numărului **a** şi este diferit de **1** şi de **a**). Câte noduri terminale (frunze) există în arbore ?

a. 5 **b.** 3 **c.** 10 **d.** 7

13. Într-un abore cu **50** noduri, numărul maxim de fii pe care poate să îi aibă un nod al său este:

a. 1 **b.** 49 **c.** 2 **d.** 0

14. Numărul minim de muchii care pot fi eliminate astfel încât graful din desenul alăturat să devină arbore este:



a. 1 **b.** 3 **c.** 2 **d.** 0

15. Care dintre următoarele şiruri de numere reprezintă gradele nodurilor unui arbore cu **5** noduri?

a. 1, 1, 3, 1, 0 **b.** 4, 1, 5, 1, 2
c. 4, 3, 2, 1, 1 **d.** 2, 1, 1, 3, 1

16. Numărul de noduri ale unui arbore cu **100** de muchii este:

a. 101 **b.** 99 **c.** 100 **d.** 50

17. Matricea de adiacenţă asociată unui arbore cu **p** noduri conţine:

a. p^2-2p+2 elemente nule **b.** **p** elemente nule
c. p^2-p elemente nule **d.** **p-1** elemente nule

18. Care este gradul maxim posibil al unui nod dintr-un arbore cu **n** noduri?

a. n-1 **b.** n DIV 2 | n/2 **c.** 2 **d.** n

19. Care dintre matricele de adiacenţă de mai jos corespunde unui arbore cu **4** noduri?

a.
```
0 0 1 1
0 0 1 0
1 1 0 1
1 0 1 0
```
b.
```
0 0 1 0
0 0 1 0
1 1 0 0
0 0 0 0
```
c.
```
0 0 1 0
0 0 0 1
1 0 0 0
0 1 0 0
```
d.
```
0 0 1 0
0 0 1 0
1 1 0 1
0 0 1 0
```

20. Se consideră arborele cu **8** noduri numerotate de la **1** la **8**, dat prin lista de muchii: **(1,2), (1,3), (3,4), (3,5), (3,6), (4,8), (4,7)**. Care dintre nodurile următoare poate fi rădăcină a acestui arbore astfel încât înălţimea lui să fie maximă (înălţimea arborelui este egală cu numărul de muchii ale celui mai lung lanţ ce uneşte rădăcina de fiecare frunză).

a. 1 **b.** 2 **c.** 4 **d.** 3

21. Un graf neorientat este graf complet dacă şi numai dacă oricare două noduri sunt adiacente. Care este numărul de muchii care trebuie eliminate dintr-un graf neorientat complet cu **8** noduri, astfel încât graful parţial obţinut să fie arbore?

a. 8 **b.** 21 **c.** 16 **d.** 20

22. Câte muchii trebuie să eliminăm dintr-un graf neorientat conex cu **12** vârfuri şi **21** de muchii astfel încât acesta să devină arbore?

a. 9 **b.** 12 **c.** 10 **d.** 11

23. Câte cicluri elementare care diferă prin cel puţin o muchie se formează prin adăugarea unei singure muchii la un arbore (ciclul este elementar dacă este format numai din noduri distincte, excepţie facând primul şi utimul)?

a. 2 **b.** 0 **c.** 1 **d.** 3

24. Se consideră matricea de adiacenţă alăturată asociată unui graf neorientat cu 7 noduri. Stabiliţi prin care dintre metodele următoare, graful dat poate deveni arbore.

```
0 1 0 1 0 0 0
1 0 1 1 1 0 0
0 1 0 1 0 0 1
1 1 1 0 0 0 0
0 1 0 0 0 0 0
0 0 0 0 0 0 0
0 0 1 0 0 0 0
```

a. eliminând două muchii şi adăugând o muchie
b. eliminând o muchie şi adăugând o muchie
c. eliminând două muchii
d. adăugând o muchie

25. Se consideră arborele cu 8 noduri şi muchiile **[1,5], [2,3], [3,6], [3,8], [4,6], [5,7], [6,7]**. Care dintre nodurile arborelui ar putea fi alese ca radacină pentru ca arborele sa aibă număr maxim de niveluri:

a. 1, 2, 8 **b.** 3, 4, 7 **c.** 6 **d.** 5

26. Care dintre următoarele matrice este matricea de adiacenţă a unui un graf care are proprietatea că este arbore?

a.
```
0 1 1
1 0 1
1 1 0
```

b.
```
0 1 1 0 0 0
1 0 1 0 0 0
1 1 0 0 0 0
0 0 0 0 0 1
0 0 0 0 0 1
0 0 0 1 1 0
```

c.
```
0 1 1 0 0
1 0 1 0 0
1 1 0 0 0
0 0 0 0 1
0 0 0 1 0
```

d.
```
0 0 1 0 0
0 0 1 0 0
1 1 0 1 1
0 0 1 0 0
0 0 1 0 0
```

27. Se consideră un arbore cu rădăcină reprezentat în memorie cu ajutorul vectorului de taţi : `tata = (2,3,0,3,3,2,6,6,4,9)`.
Stabiliţi care dintre nodurile următoare sunt extremităţile finale ale unor lanţuri elementare de lungime impară care au ca extremitate iniţială rădăcina arborelui.

a. `10 3` **b.** `3 2 4 5` **c.** `2 4 5` **d.** `1 6 9`

28. Precizaţi câte muchii trebuie înlăturate din graful a cărui matrice de adiacenţă este dată alăturat, astfel încât să devină arbore?

```
0 1 1 1 0 0 0
1 0 0 0 1 1 0
1 0 0 1 0 0 1
1 0 1 0 0 0 0
0 1 0 0 0 0 0
0 1 0 0 0 0 1
0 0 1 0 0 1 0
```

a. `2` **b.** `1` **c.** `0` **d.** `3`

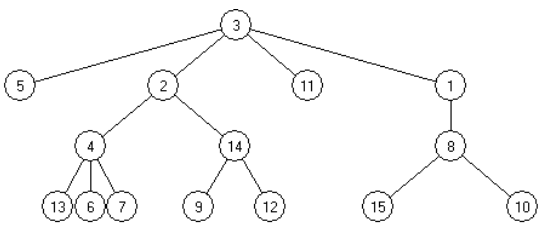
29. Într-un arbore cu exact `8` noduri rădăcina, reprezentată de nodul `1`, se află pe nivelul `1` şi fiecare nod al arborelui are cel mult `2` descendenţi direcţi. Care este înălţimea minimă posibilă pentru un astfel de arbore? (Înălţimea unui arbore=numărul maxim de muchii de la rădăcină la un vârf terminal)

a. `4` **b.** `3` **c.** `2` **d.** `1`

30. Câte lanţuri elementare de lungime maximă ce leagă două noduri ale arborelui din figura alăturată există?

a. `8` **b.** `6`

c. `10` **d.** `4`



31. Considerăm un arbore `G` cu `7` noduri care are matricea de adiacenţă alăturată. Stabiliţi care dintre următorii vectori este un vector de taţi al arborelui dat:

```
0 1 1 1 0 0 0
1 0 0 0 0 0 0
1 0 0 0 1 0 0
1 0 0 0 0 0 0
0 0 1 0 0 1 1
0 0 0 0 1 0 0
0 0 0 0 1 0 0
```

a. `(0,1,1,1,3,5,5)` **b.** `(0,1,3,1,1,5,5)`
c. `(0,1,5,5,3,3,5)` **d.** `(0,1,1,1,5,3,3)`

32. Un arbore cu rădăcină are `n` noduri numerotate de la `1` la `n`. Dacă vectorul de taţi al acestui arbore (vector notat în continuare cu `t`) are proprietatea că

`t[i]=i-1` pentru `i = 1,2,...,n`

atunci numărul de noduri care au exact un descendent direct în acest arbore este egal cu:

a. `0` **b.** `n-1` **c.** `n` **d.** `1`

33. Fie arborele **G=(V,E)** în care mulţimea vârfurilor este **V={1,2,3,4,5,6,7,8,9,10}**, iar mulţimea muchiilor este **E={[1,3],[1,4],[2,1],[2,5],[3,7],[4,8],[4,9],[5,6],[9,10]}**. Considerând vârful **1** rădăcina arborelui, vectorul de taţi corespunzător arborelui **G** este:

a. **T=(0,1,1,3,1,5,3,4,9,4)** **b.** **T=(0,1,1,1,3,5,3,4,4,4)**
c. **T=(0,1,1,1,5,2,4,3,4,9)** **d.** **T=(0,1,1,1,2,5,3,4,4,9)**

34. Un arbore cu **9** noduri, numerotate de la **1** la **9**, este memorat cu ajutorul vectorului de taţi **t=(2,5,5,3,0,2,4,6,6)**. Ascendenţii nodului **6** sunt:

a. **1** şi **4** **b.** **2** **c.** **8** şi **9** **d.** **2** şi **5**

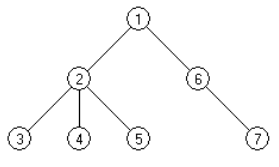
35. Într-un arbore reprezentat prin vectorul de taţi **t:(8,8,0,3,4,3,4,7)**, numărul descendenţilor nodului **4** este egal cu:

a. **7** **b.** **2** **c.** **5** **d.** **3**

36. Un arbore cu nodurile numerotate de la **1** la **9**, este memorat cu ajutorul vectorului de taţi **(2,5,5,3,0,2,3,7,6),** atunci nodurile frunză ale arborelui sunt:

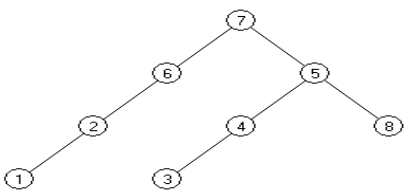
a. **6,7** **b.** **1,4,8,9** **c.** **5** **d.** **2,3**

37. Stabiliţi care dintre următorii vectori este vector de taţi pentru arborele cu rădăcina **1** din figura alăturată:



a. **1 1 2 2 3 1 6** **b.** **0 1 2 2 4 1 6** **c.** **0 1 2 2 2 1 6** **d.** **0 1 2 3 4 5 6**

38. Stabiliţi care dintre următorii vectori este vector de taţi pentru arborele cu rădăcina **7** din figura alăturată.



a. **2 6 4 5 7 7 0 5** **b.** **1 2 4 5 6 7 0 3**
c. **2 6 3 5 7 7 0 5** **d.** **2 6 7 3 4 5 0 8**

39. Pentru reprezentarea unui arbore cu **8** noduri, numerotate cu numere de la **1** la **8**, se utilizează vectorul de taţi **TATA =(3,4,7,7,4,7,0,5).** Care sunt frunzele arborelui?

a. **1,2,3,8** **b.** **3,4,5,7** **c.** **1,2,6,8** **d.** **1,2,3,4**

40. Un arbore cu rădăcină cu **9** noduri are vectorul tată **TATA=(6,6,0,3,3,3,4,4,3**). Numărul nodurilor sale terminale este:

a. **5** **b.** **6** **c.** **4** **d.** **3**

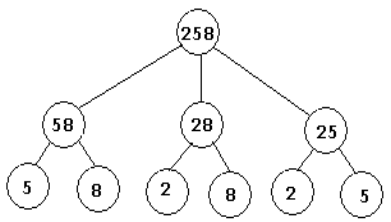
41. Se consideră un arbore cu **10** noduri dat prin următorul vector **Tata=(3,3,0,3,2,2,5,5,4,6).** Care sunt nodurile terminale ale arborelui?

a. 7 8 **b.** 9 10 **c.** 1 7 10 **d.** 1 7 8 9 10

42. Fie un arbore cu **7** vârfuri, etichetate cu numere de la **1** la **7**, dat prin vectorul **Tata=(7,7,1,1,1,2,0).** Să se precizeze care este rădacina arborelui.

a. 2 **b.** 6 **c.** 3 **d.** 7

43. Se consideră un arbore cu rădăcină în care orice nod care nu este rădăcină memorează un număr obţinut prin ştergerea unei cifre din numărul păstrat în nodul tată (conform exemplului din figura alăturată). Ştiind că rădăcina memorează valoarea **1234**, că fiii oricărui nod sunt diferiţi şi că orice frunză conţine o singură cifră, stabiliţi câte frunze memorează cifra **1**.



a. 6 **b.** 12 **c.** 3 **d.** 1

44. Se consideră arborele cu **8** noduri, numerotate de la **1** la **8**, dat prin lista de muchii: **(1,2), (1,3), (3,4), (3,5), (3,6), (4,8), (4,7)**. Dacă alegem ca rădăcină a arborelui nodul **3**, atunci vectorul de taţi corespunzător arborelui este:

a. (0,1,1,3,3,3,4,4) **b.** (2,3,0,3,4,5,6,7)
c. (2,3,0,7,3,3,4,1) **d.** (3,1,0,3,3,3,4,4)

45. Un arbore are **10** noduri. Care este numărul maxim de cicluri elementare distincte care se pot forma dacă în arbore adăugăm două muchii distincte?

a. 2 **b.** 3 **c.** 1 **d.** 4

46. Fie un arbore precizat prin vectorul de taţi **T=(0,1,2,5,2,8,8,2)**. Care este numărul maxim de descendeţi direcţi ai unui nod din arbore?

a. 3 **b.** 0 **c.** 2 **d.** 1

47. Care din următorii vectori **NU** poate fi vectorul de taţi pentru un arbore cu **6** noduri?

a. T=[3,3,0,3,3,3] **b.** T=[2,0,1,2,3,4]
c. T=[0,1,5,1,3,2] **d.** T=[2,3,4,5,6,0]

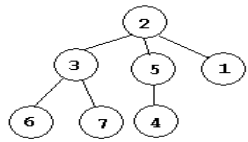
48. Se consideră arborele cu **18** noduri având nodurile numerotate de la **1** la **18** şi vectorul de taţi **(12,17,4,0,12,17,13,1,14,13, 14,3,16,4,17,14,3,6)**. Considerând că rădăcina arborelui se află pe nivelul **1**, stabiliţi câte noduri se află pe nivelul **3**.

a. 4 **b.** 5 **c.** 3 **d.** 6

49. Un arbore cu rădăcină are nodurile numerotate de la 1 la 5. Care dintre următorii vectori nu poate fi vector de taţi?

a. `2 0 1 1 2` **b.** `4 1 1 0 2` **c.** `3 4 0 2 3` **d.** `3 1 0 1 2`

50. Pentru arborele cu rădăcină din figura alăturată vectorul de "taţi" este:



a. `0 5 7 4 0 0 3` **b.** `0 5 7 0 4 3 3`

c. `2 0 2 5 5 3 3` **d.** `2 0 2 5 2 3 3`

51. Pentru care dintre următorii arbori cu rădăcină, memoraţi cu ajutorul vectorilor de taţi, nodurile **4**, **6** şi **9** sunt singurii descendenţi direcţi ai nodului **3**?

a. `tata=(3,3,4,0,2,3,4,4,4)` **b.** `tata=(6,4,9,0,3,3,3,3,3)`

c. `tata=(2,0,2,3,2,3,4,4,3)` **d.** `tata=(0,3,1,3,2,3,4,4,3)`

52. Într-un arbore binar (un arbore binar este un arbore în care fiecare nod are cel mult doi descendenţi direcţi), un lanţ care uneşte rădăcina cu oricare din nodurile frunză, conţine cel mult **n-1** muchii. Care este numărul maxim de noduri dintr-un astfel de arbore?

a. 2^n-1 **b.** `n` **c.** `2n` **d.** 2^{n-1}

53. Se consideră arborele cu rădăcină dat prin vectorul de taţi **t=(5,7,5,7,7,9,0,9,4,3,5,11,4,4,4)**. Câte lanţuri de lungime **2**, care pornesc din rădăcină există?

a. `7` **b.** `11` **c.** `4` **d.** `14`

54. Care dintre următorii vectori poate reprezenta vectorul de taţi al unui arbore cu rădăcină?

a. `(5,7,1,1,0,7,7,12,1,12,4,7)` **b.** `(5,7,1,1,0,7,0,12,1,12,4,7)`

c. `(5,7,1,1,0,7,5,12,1,12,4,7)` **d.** `(0,7,1,1,8,7,5,12,1,12,4,7)`

55. Se consideră arborele cu **14** noduri având următoarele muchii: **[3,4], [4,14], [14,13], [4,5], [1,5], [5,7], [2,7], [6,7], [6,9], [8,9], [9,12], [11,12], [10,12]**. Care dintre vectorii următori reprezintă vectorul de taţi al arborelui dat?

a. `(5,7,4,5,0,7,5,9,6,12,12,11,14,4)`

b. `(5,7,4,0,4,7,5,9,6,0,12,9,14,4)`

c. `(0,7,4,5,1,7,5,9,6,11,12,9,14,4)`

d. `(5,7,4,5,7,9,6,9,12,12,12,0,14,4)`

56. Pentru reprezentarea unui arbore cu rădăcină cu **9** noduri, etichetate cu numere de la **1** la **9**, se utilizează vectorul de taţi **TATA =(4, 1, 1, 0, 1, 3, 3, 7, 4)**. Care sunt frunzele arborelui?

a. `2,5,6,8,9` **b.** `1,4,6,8,9` **c.** `2,3,4,5,6` **d.** `2,6,7,8,9`

57. Se consideră vectorul de taţi al unui arbore oarecare **t=(0,3,1,3,1)**, în care nodurile sunt numerotate cu **1,2,3,4,5**. Alegeţi **afirmaţia incorectă** :

a. nodurile **3** şi **5** sunt fraţi
b. nodul **1** este rădăcină
c. nodul **3** este fiul nodului **2**
d. nodurile **2,4,5** sunt frunze

58. Se consideră vectorul de taţi al unui arbore oarecare **t=(0,3,1,3,1,5)**, în care nodurile sunt numerotate de la **1** la **6**. Alegeţi afirmaţia corectă:

a. nodurile **2**, **4**, **6** sunt fraţi
b. nodul **5** are gradul **1**
c. nodul **3** este tatăl nodului **1**
d. nodurile **2**, **4** şi **6** sunt frunze

59. Un arbore cu rădăcină are nodurile numerotate de la **1** la **5**. Care dintre următorii vectori poate fi vector de taţi?

a. **4 4 1 0 1** **b.** **4 4 1 2 1** **c.** **2 3 0 4 3** **d.** **1 2 0 3 4**

60. Se consideră arborele dat prin vectorul tata: **t=(3,3,8,8,8,5,8,0,3,3)**. Câte lanţuri elementare de lungime **2**, care pornesc din rădăcină există în arbore ?

a. **4** **b.** **7** **c.** **6** **d.** **5**

61. Se consideră arborele format din **9** noduri numerotate de la **1** la **9**, dat prin vectorul de taţi **t=(5,5,2,2,0,5,9,9,5)**. Câte lanţuri distincte de lungime **3** care au ca extremităţi noduri terminale (frunze) există? Lanţul de lungime **3** **(6,5,9,7)** se consideră identic cu lanţul **(7,9,5,6)**

a. **8** **b.** **2** **c.** **5** **d.** **4**

62. Pentru un arbore cu **9** noduri, care dintre următorii vectori ar putea fi vector de taţi?

a. **(4,3,0,3,9,9,6,6,9)** **b.** **(4,3,0,3,9,9,6,6,3)**
c. **(4,3,2,3,9,9,6,6,3)** **d.** **(4,3,2,3,9,9,6,6,0)**

63. Se consideră un arbore cu rădăcină reprezentat în memorie cu ajutorul vectorului de taţi : **tata=(2,3,0,3,3,2,6,6,4,9)**. Stabiliţi care dintre nodurile arborelui sunt extremităţile finale ale unor lanţuri elementare de lungime **3** care au ca extremitate iniţială rădăcina arborelui.

a. **7 8 10** **b.** **1 6 9** **c.** **4 5 6** **d.** **2 4 5**

64. Un arbore are nodurile numerotate cu numere distincte de la **1** la **5**. Vectorul de taţi asociat arborelui poate fi :

a. **2, 1, 0, 3, 4** **b.** **2, 4, 0, 3, 4**
c. **5, 4, 2, 1, 3** **d.** **5, 2, 4, 5, 0**

65. Care dintre următorii vectori **"de taţi"** corespunde reprezentării unui arbore în care nodurile numerotate cu **6, 4** şi **9** sunt descendenţi direcţi ai nodului **3**?

a. **tata=(3,3,4,0,2,3,4,4,4)** **b.** **tata=(9,9,4,9,9,9,9,9,0)**
c. **tata=(3,3,1,3,2,3,4,4,3)** **d.** **tata=(3,0,2,3,2,3,4,4,3)**

Indicaţii şi rezolvări

1.1.	1.c	2.d	3.b	4.b	5.a	6.a
	7.a	8.a	9.c	10.b	11.b	12.a
	13.a	14.c	15.a	16.a	17.d	18.a
	19.d	20.d	21.c	22.c	23.a	24.d
	25.d	26.d	27.c	28.a	29.d	30.a
1.2.	1.c	2.a	3.c	4.d	5.b	6.c
	7.c	8.b	9.c	10.c	11.a	12.d
	13.c	14.d	15.a	16.c	17.a	18.a
	19.c	20.c	21.a	22.d	23.c	24.d
	25.b	26.a	27.d	28.c	29.d	30.d
	31.d	32.c	33.a	34.d	35.c	36.b
	37.c	38.b	39.c	40.c	41.a	42.a
	43.c	44.c	45.d	46.b	47.a	48.d
	49.a	50.a	51.a	52.a	53.c	54.c
	55.a	56.d	57.a	58.c	59.a	60.b
	61.d	62.c	63.b	64.c	65.c	66.a
	67.c	68.d	69.b	70.d	71.b	72.d
	73.d	74.b	75.a	76.b	77.b	78.c
	79.c	80.d	81.c	82.c	83.a	84.a
	85.d	86.b	87.a	88.b	89.c	90.d
	91.c	92.b	93.a	94.b	95.b	96.b
	97.c	98.d	99.b	100.a	101.a	102.b
	103.a	104.b	105.c	106.a	107.c	108.c
	109.c	110.c	111.a	112.d	113.b	114.b
	115.c	116.d	117.a	118.d	119.c	120.c
	121.c	122.d	123.b	124.d	125.c	126.c
	127.b	128.a	129.d	130.b	131.d	132.d

1.3.

1. 1) 25. 2) (numărul de numere citite **b** din intervalul **[10,20]**, cu proprietatea că **cmmdc(b,15)=1**)

2. 1) 6. 2) Structura **cât timp...execută** poate fi înlocuită cu structura **repetă...până cÂnd.**

```
┌repetă
│   a←a-b; k←k+2
└până când a<b
```

3.

1) 5

3) Structura **cât timp...execută** poate fi înlocuită cu structura **repetă...cât timp.**

4) 0 şi 1

4) 4 perechi de numere **(1,5) (5,1) (2,10) (10,2)** au proprietatea că împărţind numărul mai mare la cel mai mic se obţine câtul **5**.

```
┌dacă a≤b atunci
│┌repetă
│| a←a+2; s←s+1
│└■cât timp a≤b
└■
```

4.

1) 0 şi 5

2) Orice valoare mai mică strict decât **5**. De exemplu **3.**

4) Structura **cât timp...execută** se poate înlocui cu structura **repetă...cât timp.**

```
┌dacă i<n atunci
│┌repetă
│| i←i+1; x←p*i
│| p←x mod 10
│└■ cât timp i<n şi p<>0
└■
```

5. 1) NU. 2) Dacă numărul de cifre ale lui **b** este mai mic sau egal decât numărul cifrelor lui **a** şi dacă fiecare cifră a lui **a** este cel puţin egală cu cifra aflată pe aceeaşi poziţie în **b**, atunci se va afişa mesajul **DA**. De exemplu **a=1234** şi **b=123**

4) Structura **cât timp...execută** se poate înlocui cu structura **repetă...cât timp.**

```
┌dacă (a>0)şi(b>0) atunci
│┌repetă
│││┌dacă (a mod 10)<(b mod 10) atunci x←0
││└■
│└■ a←[a/10]; b←[b/10]
└■ cât timp (a>0)şi(b>0)
```

6. 1) NU. 2) Dacă numărul citit are prima şi ultima cifră de aceeaşi paritate atunci se va afişa mesajul **DA**. De exemplu **12345.**

4) Structura **cât timp...execută** se poate înlocui cu structura **repetă...cât timp.**

```
┌dacă n>=10 atunci
│┌repetă n← [n/10]
│└■cât timp n>=10
└■
```

7. 1) 2 5. 2) Unul dintre răspunsurile corecte: **10,20,30,40,50, 10,20,30,40, 100,0.**

4)

```
citeşte x {x natural}
nr←0; s←0
┌daca x≠0 atunci
│ ┌repeta
│ │ nr←nr+1
│ │┌dacă nr%2=0 atunci  s←s+x%10
│ │└■
│ │  citeşte x
│ └pana cand x=0
└■
scrie s,nr
```

8.

1) 13479

2) 31

3) Putem înlocui structura **cât timp ... execută** cu structura **repetă ... cât timp** sau cu structura **repetă ... până când**.

```
┌repetă
│ ┌dacă n%2>0 atunci scrie i
│ └■
│ i←i+1
│ n←[n/2]
└■cât timp când n≠0
```

9. 1) -512. 2) Algoritmul calculează x^m şi dacă **x=1** atunci **m** poate fi orice număr natural iar dacă **x≠1** atunci **m=0.**

4)

```
citeşte x ,m
y←1
┌ daca m>0 atunci
│┌ repeta
│┌ dacă m%2 =0 atunci m←[m/2]; x←x*x
│└■              altfel m←m-1;y←y*x
│└■
└pana cand m=0
└■
scrie y
```

10. 1) 105. 2) n=14,k=13.

4) Algoritmul calculează şi afişează suma multiplilor numărului **k**, multipli situaţi în intervalul **[1;n]**. Fie x numărul acestora **(x=[n/k])**. Suma multiplilor este **s=1*k+2*k+…+x*k=k(1+2+…+x)=k*x*(x+1)/2.**

```
var n,k,x:word;
begin
readln(n,k); x:=n div k;
write(k*x*(x+1) div 2)
end.
```

```
# include <iostream.h>
void main()
{ cin>>n>>k;
  x=n/k;
  cout<< k*x*(x+1)/2;}
```

11. 1) 0. 2) 2358. 3) 54. Se afişează **1** pentru numerele ce au cifrele în ordine crescătoare (de la stânga la dreapta) : **1, 2,...9, 11...19, 22...29, 33...39,...., 88, 89, 99** adică **9+9+8+7+6+5+4+3+2+1=54** numere.

12. 1) 67. 2) 9(9+4+2+1). 4) 3 (pentru **n=1** se afişează **1**, pentru **n=2** se afişează **3** iar şirul valorilor afişate este un şir crescător funcţie de **n)**

13.

1) 49

2) (n,m)∈{(5,0),(5,1),(6,4),(8,7)}

3) Secvenţa alăturată reprezintă o posibilitate de înlocuire.

```
dacă n≥m atunci
  repetă
    s←s+n; n←n-1
  cât timp n≥m
```

14. 1) 1. 2) Orice număr natural de minim trei cifre, cifrele nefiind în ordine descrescătoare sau strict crescătoare. De exemplu **324.**

4) Structura **cât timp...execută** se poate înlocui cu structura **repetă...cât timp.**

```
dacă n>9 atunci
  repetă
    dacă n%10>[n/10]%10 atunci y←1
       altfel y←0
    dacă x ≠ y atunci ok←0
    n←[n/10]
  cât timp n>9
```

15. 1) 14. 2) n=6 sau orice numar natural care se decompune în produs de numere prime distincte

16. 1) 2. 2) n=3533 şi **c=5**

4) Putem înlocui structura **cât timp ... execută** cu structura **repetă ... cât timp** sau cu structura **repetă ... până când**.

```
dacă n%10=c atunci
   repetă
     n←[n/10]
     k←k+1
   până când n%10≠c
```

17.

1) 6

2) De exemplu **2 5 0**

4) Putem înlocui structura **cât timp ... execută** cu structura **repetă ... cât timp** sau cu structura **repetă ... până când.**

```
dacă a≠0
  repetă
  citeşte b
    dacă a%2=b%2 atunci
k←k+1
  a←b
  până când a=0
```

18. **1)** **2** **8**

2) De exemplu **1000**

4) Putem înlocui structura **cât timp ... execută** cu structura **repetă ... cât timp** sau cu structura **repetă ... până când**

```
┌ repetă
│    b←b*10
│    c←c+1
└■ cât timp b<1
```

19. **1)** **6**

2) De exemplu: **10090** şi **55045** sau orice pereche de două numere naturale cu **5** cifre şi care au cifra de control **1**

4) Putem înlocui structura **cât timp ... execută** cu structura **repetă ... cât timp** sau cu structura **repetă ... până când**

```
┌repetă
│ s←0
│┌repetă
│└  s←s+n%10; n←[n/10]
└■ până când n=0
  n←s
└■ cât timp n>=10
```

20.

1) **26904**

2) **99**

4) Algoritmul echivalent este cel alăturat.

```
citeşte n  (număr natural, nenul)
p ← 1
┌dacă n>1 atunci
│┌repetă
└└ n ← ([n/p]+1)*p+n%p; p ← p*10
 └■până când p≥n
 ■scrie n
```

21.

1) **1,2,-1** şi **-1,3,-2**

2) De exemplu **1 5 6 0**

3) Putem înlocui structura **cât timp ... execută** cu structura **repetă ... cât timp** sau cu structura **repetă ... până când**

```
┌repetă
│ d←b*b-4*a*c
│  ┌dacă d>0 atunci
│  │ scrie a,b,c
│  └■
│  a←b; b←c; citeşte c
└■ cât timp c<>0
```

22.

1) **3**

2) **108**

3) Putem înlocui structura **cât timp ... execută** cu structura **repetă ... cât timp** sau cu structura **repetă ... până când**

```
┌dacă n<>m
│┌repetă
└└ ┌dacă n>m atunci n ← n-m
  └          altfel m ← m-n
   └■
  └■ până când n=m
  ■
```

23. **1)** **3.** **2)** orice număr de cel puţin două cifre care are suma cifrelor mai mică sau egală cu **9**. **3)** **9992**

24. **1)** **82**

2) De exemplu **n=715** sau orice alt număr natural format doar din cifre impare.

3) Putem înlocui structura **cât timp** ... **execută** cu structura **repetă ... cât timp** sau cu structura **repetă ... până când**

```
┌ repetă
│  c ←n%10; n←[n/10]
│   ┌ dacă c%2=0
│   │ atunci z←z*10+c
│   └■
└■ până când n=0
```

25. **1)** **5.** **2)** Oricare dintre numerele **16,32,64.** Valorile afişate sunt **divizori** ai numărului **n**. Rezultă de aici că numărul este par şi că valoarea lui **d** este **2** (primul divizor găsit). Pentru afişarea valorii variabilei **d(=2)** trebuie ca **n** să ajungă egal cu **1**, ceea ce presupune ca $n=2^k$. Puterile lui **2** , numere cu **2** cifre, sunt **16,32** şi **64.**

3) **0 numere**

26. Algoritmul construieşte numărul format din cifrele impare ale numărului a, aşezate în ordinea inversă apariţiei lor în cadrul numărului a.

1) 971. 2) 8851

3) 15 (numerele care conţin doar cifre pare din intervalul **[0,50]** adică **0, 2, 4, 6, 8, 20, 22, 24, 26, 28, 40, 42, 44, 46, 48).**

27. Algoritmul determină cel mai mic număr din şirul lui Fibonacci **(1, 1, 3, 5, 8, 13,** …) care este mai mare decât n.

1) 21. 2) 21. 3) 5 (9, 10, 11, 12, 13)

28. Algoritmul afişează **1** dacă suma cifrelor pare din **n** este egală cu suma cifrelor impare, şi respective **0** în caz contrar.

1) 0

2) Există două perechi: **(2,0), (0,2)**

4) Algoritmul echivalent foloseşte structura **repetă...până când**, în locul structurii **cât timp**.

```
citeşte n (nr. natural)
s1 ← 0; s2 ← 0
nr ← 0
┌repetă
│ ┌dacă n % 2 = 0 atunci s1 ← s1 + n % 10
│ │  altfel s2 ←s2 + n % 10
│ └■
│ n ← [n/10]
└până când n=0
┌dacă s1 = s2 atunci  nr ← 1
└■
scrie nr
```

29. Algoritmul determină cea mai mare cifră dintre cifrele numărului **[n/10].**

1) 7

2) 40 (numerele cuprinse între **120** şi **129** respectiv **200** şi **229)**

3) Algoritmul echivalent este cel alăturat.

```
citeşte n {nr. natural}
max ← 0
┌repetă
│ n ← [n/10]
│ ┌dacă max<n%10 atunci
│ │  max ←n%10
│ └■
└până când n=0
scrie max
```

30. 1) 0. 2) oricare dintre numerele de forma **11*k, 1≤k≤9. 4) 9**

31. 1) 10. 2) 6 perechi: **(1,5) (2,6) (3,7) (4,8) (5,9) (6,10)**

4) Algoritmul calculează numărul de numere naturale din intervalul **[x;y]**

```
citeste x,y; scrie y-x+1
```

32. 1) 512. 2) 4

4) structura **cât timp** … **execută** va fi inlocuită cu

```
┌cât timp y > 0 execută
│    y ← y – 1
│    p ← p * x
└■
```

33. 1) 1. 2) 45. 3) i←i-1

34. Algoritmul determineă cea mai mare cifră din scrierea numărului **x** precum şi de câte ori apare această cifră în scrierea lui **x**.

1) se afişează **8 2**

2) Numărul x trebuie să aibă cifra maximă egală cu **5**, şi să apară de **3 ori.** Un exemplu de astfel de număr este: **152535.**

35. 1) NU. 2) 510

4) Putem înlocui structura **cât timp** ... **execută** cu structura **repetă ... cât timp** sau cu structura **repetă ... până când**

```
┌ repetă
│   a←[a/10]
└■cât timp a>b
```

36. 1) 0 (se construieşte în **c** numărul **1111;** se scad succesiv din **1234** numerele **1111,111,11,1). 2) 13 (13-11-1=1). 3) 3** numere **(1,12,123)**

37. **1)** 32

3)
```
┌dacă n>9 atunci
│  ┌repetă
│  │  n←[n/10]; b←n%10
│  │ ┌dacă a>b atunci m←m*10+b; a←b
│  │ └■
│  └■ până când n≤9
└■
```

4) orice număr care are ultima cifră egală cu **0**

38. **1)** **2.** **2)** **30.** **4)** Structura **cât timp**…**execută** se poate înlocui cu structura repetitivă **repetă...până când** sau **repetă...cât timp.**

39. **1)** **6 şi 12.** **2)** Orice valoare **a** pentru care **6** este multiplul lui **a.**
De exemplu **a=2** sau **a=6**, etc.

4)
```
citeşte a; k←a+5+6%a; b←6%a;  scrie b,k
```

40.

1) **52, 72, 51, 4, 69**

2) **3, 1427, 41237, 421357**

4) Algoritmul echivalent este ca cel alăturat şi foloseşte ciclul **pentru** în locul ciclului **cât timp**.

```
citeşte n (nr. natural)
┌pentru i=1,n execută
│ citeşte x (nr. natural)
│  nr←0
│ ┌cât timp x>0 execută
│ │      nr ←nr*10+x%10;  x ←
[x/1000]
│ └■
│  scrie nr
└■
```

41. **1)** **0**

2) **325 şi 325** (oricare două numere naturale egale; se afişează valoarea **1** dacă **a** şi **b** au acelaşi cifre în aceleaşi poziţii,adică a=b şi **0** în caz contrar.)

4) O posibilă soluţie este dată programul pseudocod următor:

```
citeşte a,b
┌dacă a=b atunci scrie 1
│  altfel scrie 0
└■
```

42. **1)** **11** (Algoritmul afişează cel mai mare divizor prim al numărului citit)

2) **105** (**105 = 3*5*7**)

4) Un număr este prim dacă şi numai dacă cel mai mare divizor prim a lui este chiar el.
Se înlocuieşte instrucţiunea **scrie m** cu secvenţa:

```
┌dacă m=n atunci scrie „DA”
│ altfel scrie „NU”
└■
```

43. Algoritmul calculează suma cifrelor pare dintre ultimele k cifre ale numărului n.

1) **6**

2) orice număr având ultimele **3** cifre impare sau **0**, de exemplu **126000, 1357** etc.

4) programul pseudocod alăturat este echivalent cu cel dat:

```
citeşte n,k
(n,k numere naturale)
s←0
┌pentru i←1,k execută
│ c←n%10
│ ┌dacă c%2=0 atunci s←s+c
│ └■
│  n←[n/10]
└■
scrie s
```

44.

1) 1

2) Orice şir de 5 numere toate impare

3) Structura

```
┌dacă d>m atunci m←d
└■
```

se va înlocui cu structura

```
┌dacă d=0 atunci
│   m←m+1
└■
```

45. 1) 7. **3)** 8. **4)** Oricare trei valori alese pentru **n**, astfel încât suma puterilor factorilor primi să fie aceeaşi. De exemplu : $1500=5^3*2^2*3$, $64=2^6$, $53361=7^2*11^2*3^2$

46. Algoritmul construieşte numărul foprmat din prima cifră a fiecărui număr citit de la tastatură, în ordinea inversă a citirii acestor numere.

1) 3145

2) Se citesc 3 numere a căror primă cifră este în ordine 3, 2, şi respectiv 1 apoi valoarea 0. De exemplu 3514, 261, 1224, 0.

4) Se înlocuieşte structura **cât timp** cu structura **repetă...până când** ca în algoritmul alăturat.

```
citeste n {n nr natural}
a←0; p←1
┌dacă n≠0 atunci
| ┌repetă
| | ┌dacă n>9 atunci
| | | ┌repetă
| | | | n←[n/10]
| | | └până când n<=9
| | └■
| | a←n*p+a; p←p*10
| | citeşte n
| └până când n=0
└■
scrie a
```

47. Algoritmul afişează numărul de numere din intervalul [a,b] care sunt formate doar din cifre pare.

1) 3 (numerele 204, 206, 208)

2) Numerele care se vor număra sunt 24, 26, 28, 40, aşadar b poate fi 40 sau 41.

4) 31, 36 sau orice pereche de numere între care nu există numere compuse doar din cifre pare

48. Algoritmul determină frecvenţa de apariţie a celei mai mari dintre cifrele unităţilor ale celor n numere citite de la tastatură.

1) 3

2) 28, 216, 14, 226, 6

4) Algoritmul echivalent, prezentat alăturat, foloseşte structură **cât timp**.

```
citeşte n (nr. natural)
nr ← 0; m ← 0; i←1
┌cât timp i≤n execută
|  citeşte x (nr. natural)
|  cif←x%10
|  ┌dacă cif>m
|  | atunci m ← cif; nr ← 1
|  | altfel ┌dacă cif=m atunci nr←nr+1
|  |        └■
|  └■
|  i←i+1
└■
scrie nr
```

49. Algoritmul determină cel mai mare divizor comun al numerelor a şi b.

1) 6

2) 98

3) Se va folosi structura cât timp în locul structurii pentru ca în algoritmul alăturat.

```
citeşte a,b (numere naturale)
c←0; i←1
┌cât timp i≤a execută
|┌dacă b%i=0 atunci┌dacă a%i=0 atunci
c←i
||                └■
|└■
|i←i+1
└■
┌dacă c>0 atunci scrie c
└■
```

50. Algoritmul determină câte numere prime se găsesc în intervalul **[n,m]**.

1) 2

2) 14

3) 4 (pentru **m=13, 14, 15, 16** se afişează valoarea 2)

51.

1) 5

2) a=11, b=14

4) Algoritmul calculează şi afişează numărul multiplilor lui n din intervalul **[a;b]**. Acest număr este egal cu **[b/n]-⌈a/n⌉+1**.

Secvenţa pseudocod alăturată reprezintă o soluţie pentru obţinerea valorii cerute.

```
citeşte n,a,b
s←[b/n]-[a/n]
┌daca a%n =0
│    atunci s←s+1
└■
scrie s
```

52.

1) 001

2) 8

4) La fiecare pas se calculează în **p** factorialul numărului **i** şi se afişează rezultatul obţinut în urma evaluării expresiei **n!/(2*i)=1*2*...*i/(2*i).**

```
citeşte n
    (număr natural)
 p←1
┌pentru i←1,n execută
│ p←p*i; scrie [p/(2*i)]
└■
```

53. **1)** 9. **2)** 34

4) citeste n; scrie n-1

54.

1) 21

3) Algoritmul echivalent este cel alăturat

4) Se observă că algoritmul calculează suma

1+2+...+(n-1) = n*(n-1)/2

```
citeşte n {n ∈ N*}
S←0; i←1
┌cât timp i≤n-1 execută
│  j←i+1
│┌cât timp j≤n execută
│|    S←S+1; j←j+1
│└■
│  i←i+1
└■
scrie S
```

55.

1) 8

2) De exemplu, **20 30 14 9** (Oricare patru numere naturale care nu au nici un divizor propriu comun).

4) Putem înlocui structura **repetă ... până când** cu structura **cât timp** ... **execută**

```
┌ cât timp x % d <> 0 execută
│  r ← x % d; x ← d; d ← r
└■
```

56. **1)** 54321. **2)** De exemplu, pot fi citite numerele **6921, 532, 83, 4** (primul număr format din 4 cifre cu ultima cifră 1, al doilea de 3 cifre cu ultima cifră 2, etc…).

4) Putem înlocui structura **cât timp** ... **execută** cu structura **repetă ... cât timp** sau cu structura **repetă ... până când**

```
┌dacă x>9 atunci
│┌repetă
||  nr ← nr*10; x ← [x/10]
│└până când x<=9
└■
```

57.

1) **7**

2) **1114** (Sau orice număr natural în de patru cifre în care cifra cea mai mare este 4)

3)
```
citeşte n
k←0
┌cât timp n≠0 execută
│  dacă n mod 10 > n atunci k←n mod 10
│  n ← n div 10
└■
scrie k
```

58.

1) **53**

2) **1** şi **3**, **3** şi **6**, **6** şi **9**

4) Programul calculează valoarea obţinută prin însumarea numerelor care nu sunt divizibile cu **3** din intervalul **[a,b]** şi scăderea numerelor din respectivul interval care sunt divizibile cu **3:**

```
x←[a/3]
┌dacă a%3 ≠0 atunci x←x+1
└■
y←[b/3]; S← (a+b)*(b-a+1)/2 – 3*(x+y)(y-x+1); scrie S
```

Dacă notăm cu f_n=1+2-3+4+5-6...+/-**n** (+ sau – în funcţie de divizibilitatea lui **n** la **3**), atunci rezultatul algoritmului este $f_b - f_{a-1}$. Pentru a calcula f_n adunăm şi scădem termenii negativi din expresie, obţinând diferenţa între suma termenilor a două progresii aritmetice.

O altă posibilitate de a rezolva această cerinţă este să implementăm o funcţie recursivă care să calculeze valoarea expresiei.

Iată şi un exemplu de implementare pentru o astfel de funcţie care va fi apeltată prin f(a,b):

```
function f(i, b:integer):integer;
begin
 if i=b+1 then f:=0
  else
   if i mod 3 = 0
     then f:=f(i+1,b)-i
     else f:= f(i+1,b)+i
end;
```

```
int f(int i, int b)
{ if (i==b+1) return 0;
   else
   if (i%3==0)
      return f(i+1,b)-i;
      else
         return f(i+1,b)+i;
}
```

59.

1) 3 4 5

2) 12 (se afişează triplete de numere pitagorice. Dacă **n≥13**, algoritmul afişează atât tripletele **3,4,5** şi **6,8,10** cât şi tripletul **5,12,13.**)

4) O posibilă soluţie este dată programul pseudocod următor:

```
citeşte n {n∈N}
┌pentru i←1,n-2 execută
│ ┌pentru j←i+1,n-1 execută
│ │  k← √(i*i+j*j)
│ │ ┌dacă (k<=n) şi ([k]=k) atunci scrie i,j,k
│ │ │
│ │ └■
│ └■
└■
```

unde prin **[k]** s-a notat partea întreagă a numărului **k**

60.

1) 3

3) Structura **pentru** ... **execută** poate fi inlocuită cu structura **cât timp** ... **execută**

```
i←x
┌cât timp i<=y execută
│ ┌dacă[i/10]=i%10 atunci s←s+1
│ └■
│  i← i+1
└■
```

4) Structura repetitivă **pentru** ... **execută** poate fi înlocuită cu următoarea secvenţă:

```
┌dacă a%10<b%10 atunci s←b%10-a%10 +1
│               altfel  s←0
└■
```

61. 72. 2) 11.

3) citeşte n; s←n*(n+1); scrie s

62. 1) 5. 3) Algoritmul determină calcularea şi afişarea celui mai mare divizor comun al numerelor citite. **4)** Poate fi citită orice pereche de numere naturale prime între ele (de exemplu **12** şi **25**)

63. 1) 4. 2) Ultima cifră a numărului a^b trebuie să fie egală cu **8**. De exemplu: **a=32, b=19.**

4) Structura **pentru...execută** poate fi înlocuită cu structura **cât timp...execută** astfel:

```
i←1
┌cât timp i<b execută
│  c←c*a; c←c%10; i←i+1
└■
```

64.

1) 7

2) 3

3) Programul pseudocod alăturat reprezintă o soluţie posibilă.

```
citeşte a,n
┌dacă n%2≠0 atunci  a←a-[n/2]+3
│           altfel  a←a-[n/2]
└■
scrie a
```

65.

1) 7

2) 3

3)
```
citeşte a,n      {numere naturale}
a←a-n/2
┌dacă n%2≠0 atunci  a←a+n
└■
scrie a
```

66.

1) 3

2) 9

4) Structura **dacă...atunci** poate fi scrisă astfel

```
┌dacă s=n atunci
│ ┌pentru contor=j-1,i,-1 execută scrie contor
│ └■
└■
```

67.

1) 2

3)
```
citeşte a>b (numere naturale> a≤b)
  k←0; i←a
 ┌repetă
 │  nr←0;aux←i
 │  ┌repetă
 │  │ nr←nr*10+aux%10; aux←[aux/10]
 │  └■ până când aux=0
 │  ┌dacă nr<i atunci k←k+1
 │  └■
 │ i←i+1
 └■ până când i>b
  scrie k
```

4) De exemplu: Să se scrie un program pseudocod care să citească două numere naturale **a** şi **b** şi care să afişeze toate numerele din intervalul **[a,b]** mai mici decât răsturnatul lor. Răsturnatul unui număr este egal cu numărul obţinut din cifrele numărului iniţial citite de la dreapta la stânga.

68.

1) 23

2) Orice număr cu trei cifre în ordine strict descrescătoare (ex. **754**)

4) Variabila n devine egală cu **0** după executarea structurii repetitive numai dacă cifrele numărului **n** sunt în ordine strict descrescătoare.

```
┌dacă n=0 atunci scrie DA
│         altfel scrie NU
└■
```

69.

1) Orice şir de numere naturale ce conţine un numar egal de numere pare şi impare şi apoi **0**. De exemplu, **1 2 3 4 5 6 0**

2) De exemplu, **1 2 3 4 5 0** .

3) Putem înlocui structura **repetă ... până când** cu structura **cât timp** ... **execută**

```
citeşte x {x număr natural}
┌ cât timp x<>0 execută
│    ┌ dacă x<>0 atunci
│    │┌ dacă x%2=0
│    ││        atunci n←n+1
│    ││        altfel n←n-1
│    │└■
│    └■
citeşte x {x număr natural}
└■
```

70.

1) **A.**

2) De exemplu **51215** şi **51216** (două numere naturale consecutive în care unul din ele este un palindrom).

```
3) x←x*10+n%10
   n←[n/10]
   e←n*(n-x)*([n/10]-x)
  ┌cât timp e≠0 execută
  │ x←x*10+n%10
  │ n←[n/10]
  │ e←n*(n-x)*([n/10]-x)
  └■
```

71.

1) **897793**

2) **2789**

```
3)  x←x+(9-n%10)*p
    n←[n/10]; p←p*10
    ┌cât timp n=0 execută
    │    x←x+(9-n%10)*p
    │    n←[n/10]; p←p*10
    └■
```

72. Se observă că algoritmul determină cea mai mare valoare a lui **c** pentru care 2^c divide atât pe **a** cât şi pe **b**.

1) **2**

2) **8**, **13** (cel puţin unul dintre numere trebuie să fie impar)

4) Algoritmul echivalent este cel alăturat

```
citeşte a,b
(numere naturale nenule)
c←0; i←0; j←0
┌cât timp i*j≠1 execută
│  i←a%2; j←b%2
│ ┌dacă i+j=0 atunci c←c+1
│ └■
│  a←a*i+(1-i)*[a/2]
│  b←b*j+(1-j)*[b/2]
└■
scrie c
```

73.

1) **8**

2) Putem înlocui structura **repetă ... până când** cu structura **cât timp ... execută**

```
citeşte n (număr natural)
n←[n/10]
┌cât timp n>=10 execută
│  n←[n/10]
└■
scrie n
```

4) **10**. În urma executării algoritmului se afişează prima cifră a numărului **n** dacă acesta are cel puţin **2** cifre şi **0** în cazul în care **n** este cifră. Pentru ca rezultatul afişat să fie **3**, **n** poate fi iniţial orice număr de forma $\overline{3x}$, **x∈[0;9]**.

74. **1) 2 2 2 5. 2)** orice număr prim
4) citeşte n{n natural}

```
i←2
┌repetă
│┌ daca n%i=0 atunci scrie i; n←[n/i]
│|             altfel i←i+1
│└■
└până când n=1
```

75. Algoritmul determineă cel mai mare palindrome mai mic sau egal decât **a**.
1) 1771, 2)10. 3) 1551, sau orice palindrom de patru cifre.

76. Algoritmul elimină din numărul citit de la tastatură a cifrele de pe poziţii pare, numărând cifrele începând cu 1 de la stânga spre dreapta.
1) 237
2) 18293 sau orice alt număr de forma **1x2z3.**
3) Algoritmul echivalent, prezentat alăturat, foloseşte structura **cât timp** în locul structurii **repetă...până când**.

```
citeşte n (nr natural)
x←0
┌cât timp n>0 execută
| x←x*10+n%10
| n←[n/10]
└■
┌ cât timp x>0 execută
| n←n*10+x%10
| x←[x/100]
└■
scrie n
```

77. **1) 855.**
2) Orice valori cu proprietatea că **a*b=480**. De exemplu **a=12** şi **b=40.**
4) citeşte a,b; s←a*b; scrie s

78. **1) 8**
1) Orice număr natural cu cel puţin două cifre, prima cifră fiind egală cu **5**
2) 100 de numere

79. **1) Nu. 2) 9. 3)** Structura **pentru p←t,1,-1 execută ...** se va înlocui cu structura **repetă ...până când p=0**

80.
1) **1 2**
2) Orice număr natural cu cifre distincte
3) Structura cu test iniţial va avea forma alăturată

```
c←0
┌cât timp c <=9 execută
|...
|c←c+1
└■
```

81.
1) **1535**
2) **2399**

```
3) ┌dacă a>5  atunci a←5
   └■
   v←v+z*a
```

82.
1) **6** (Programul calculează numărul de divizori pe care îl are **n**)
2) **16** (sau oricare alta dintre următoarele valori: **25**, **36**, **49**, **64**, **81**)
4) **4** (adică numărul de numere prime din intervalul dat)

83.

1) 1

2) 101 (Sau oricare alt număr natural de trei cifre care nu e patrat perfect; Algoritmul afişează valoarea **1** dacă puterile la care apare fiecare factor prim din descompunerea lui n în factori primi sunt pare, deci afişează **1** dacă n este pătrat perfect şi **0** în caz contrar.)

4) O posibilă soluţie este dată algoritmul următor:

```
citeşte n {n∈N}
┌dacă n e pătrat perfect atunci scrie „1”
│    altfel scrie „0”
└■
```

84. Secvenţa pseudocod următoare reprezintă o soluţie a problemei. S-a notat cu **x%y** restul împărţirii numărului întreg **x** la numărul întreg **y,** cu **[z]** partea întreagă a numărului **z** şi cu **x↔y** interschimbarea valorilor conţinute de variabilele **x** şi **y.**

```
citeşte a, b
┌dacă a > b atunci a↔b
└■
┌dacă a = 0 atunci scrie „EROARE”
│           altfel scrie [b/a] şi b%a
└■
```

85. Secvenţa pseudocod următoare reprezintă o soluţie a problemei (se foloseşte faptul că cele trei numere sunt pozitive, renunţându-se la extragerea radicalului!).

```
citeşte a,b,c
┌dacă a*a = b*c sau b*b = a*c sau c*c = a*b atunci scrie „DA”
│altfel scrie „NU”
└■
```

86. Secvenţa pseudocod următoare reprezintă o posibilă soluţie a problemei.

```
citeşte a
uc ← a%10 (a%10 este restul împărţirii lui a la 10)
ok ← 1
┌cât timp a≠0 execută
│  ┌dacă a%10 ≠ uc atunci ok ← 0
│  └■
│  a ← [a/10]
└■
┌dacă ok = 1 atunci scrie „DA”
│ altfel scrie „NU”
└■
```

87. Secvenţa pseudocod următoare reprezintă o posibilă soluţie a problemei.

```
citeşte a,b,c
┌dacă (a<b)şi(b<c)atunci scrie „1”
└■
┌dacă (a>b)şi(b>c)atunci scrie „2”
└■
┌dacă (a<b)şi(b>c)atunci scrie „3”
└■
┌dacă (a>b)şi(b<c)atunci scrie „4”
└■
```

88. Secvenţa pseudocod următoare reprezintă o posibilă soluţie a problemei.

```
citeşte n
S ← 0
┌pentru i=1,n execută
│  ┌dacă i este impar atunci S ← S + √i
│  │   altfel S ← S - √i
│  └■
└■
scrie S (cu patru zecimale)
```

89.
```
citeste a
x←[a]; y←x+1; scrie x,y ( pe ecran)
```

90. Numerele fiind pare şi divizibile cu 5 rezultă că numerele afişate sunt divizibile cu 10.

```
citeşte n
┌pentru i←1,n execută
│  scrie i*10
└■
```

91. Din faptul că ultima cifră a oricărui număr de forma a^b se repetă din 4 în 4 rezultă că ultima cifră a numărului a^{2007} este aceeaşi cu ultima cifră a numărului $a^{2007\%4} = a^3$.

```
citeste a; a←a%10; scrie (a*a*a)%10
```

92.
```
citeşte x
  dacă  x=1  atunci scrie  "DA"
   altfel
    repetă
     dacă x%2=0 atunci x←[x/2]
                altfel x←[(x-3)/2]
     sf_dacă
    până când (x≤1)
  sf_dacă
 dacă  x=1  atunci scrie  "DA" altfel scrie "NU"
```

93.
```
citeşte xa,ya,xb,yb
┌dacă xa*xa+ya*ya=xb*xb+yb*yb atunci scrie "DA"
│    altfel scrie "NU"
└■
```

94. Se poate afişa o valoare **a** obţinută din înmulţirea lui **x** cu **10000** şi **b=10000.** Întrucât **x** are cel mult patru zecimale, rezultă ca **10000*x** este un număr fără zecimale, deci întreg.

95. Secvenţa pseudocod următoare reprezintă o soluţie a problemei. S-a notat cu **x%y** restul împărţirii numărului întreg **x** la numărul întreg **y** şi cu **[z]** partea întreagă a numărului **z.**

```
citeşte n; k←2
┌cât timp n≥k şi n%k=0 execută n←[n/k]; k←k+1
└■
┌dacă n=1 atunci scrie "DA", k-1
│  altfel scrie "NU"
└■
```

96. a) Ordonăm crescător valorile memorate de variabilele **a,b** şi **c** astfel încât **a≤b≤c**. Presupunem că există **[a,b] ∩ Z ≠ ∅.** Fie **A(a), B(b), C(c)** trei puncte pe axă. Fie **X(x)** un punct pe axă, **x∈Z**. Observăm că: **dist(A,X)+dist(C,X)=dist(A,C)**=constantă**.** Atunci:

dist(A,X)+dist(C,X)+dist(B,X)=dist(A,C)+dist(B,X).

Această sumă este minimă dacă **dist(B,X)** este minimă, deci pentru **|b-x|=minim**. Cum **[b]≤b<[b]+1**, deducem că **min|b-x|** este **0** dacă **b∈Z**, altfel este **min{b-[b], [b]+1-b}.** Deci: **x=[b]**, dacă **b-[b]≤[b]+1-b**; altfel **x=[b]+1.**

Dacă **[a,b] ∩ Z = ∅,** atunci este adevărată relaţia: **[a]≤a≤b≤c≤[a]+1**. Astfel alegem **x=[a]** dacă **|a-[a]|+|b-[a]|+|c-[a]|<|[a]+1-a|+|[a]+1-b|+|[a]+1-c| ⇔ 2(a+b+c)<6[a]+3**. Altfel **x=[a]+1**.

b) Secvenţa pseudocod următoare reprezintă o soluţie a problemei.

```
citeşte a,b,c
┌dacă a>b atunci d←a; a←b; b←d
└■
┌dacă a>c atunci d←a; a←c; c←d
└■
┌dacă b>c atunci d←b; b←c; c←d
└■
┌dacă c-a<1
│  atunci ┌dacă 2*(a+b+c)<6*[a]+3 atunci x←[a]
│         │   altfel x←[a]+1
│         └■
│  altfel x ←[b]
│         ┌dacă b>[b]+0.5 atunci x←x+1
│         └■
└■
scrie x
```

97. Se citesc pe rând numerele de la tastatură. La citirea unui număr pozitiv mai mic decât minimul găsit până în acel moment, atribuim minimului valoarea citită iar variabila în care contorizăm numărul de apariţii o vom reiniţializa la 1. La întâlnirea unei valori egale cu minimul provizoriu, vom incrementa contorul. În pseudocod algoritmul arată astfel:

```
min←10000; nr←0; citeşte n
┌pentru i←1,n execută
│  citeste x
│  ┌dacă x>0 şi x<min atunci min←x; nr←1
│  │  altfel ┌dacă x=min atunci nr←nr+1
│  │         └■
│  └■
└■
```

98. O soluţie posibilă verifică mai întâi dacă numărul n are cel puţin 3 cifre şi dacă ultima cifră este mai mare decât penultima.

În caz afirmativ extragem ultimele două cifre **ba** şi tăiem câte două cifre până când ulimele două cifre nu mai formează un număr egal cu **ba.**

Pentru a fi număr fierăstrău, **n** trebuie să ajungă egal cu cifra **a**.

```
citeşte n; x←n%100
┌dacă [x/10]>=x%10 sau n<=99
│ atunci scrie NU
│ altfel
│   ┌┌repetă
│   ││n←[n/100]
│   │└până când x≠n%100
│   │┌dacă n=x%10 atunci scrie DA
│   ││   altfel scrie NU
│   │└■
│   └■
└■
```

99. Observăm că şirul este format din 9 numere cu câte două cifre, 9 numere cu câte trei cifre, etc. Pentru a determina elementul de pe poziţia **n**, calculăm mai întâi câtul împărţirii numărului **n** la **9**. Fie acest număr **c**. Notăm cu **r** restul împărţirii numărului **n** la **9**. Fie acest număr **r**. Dacă **r=0** atunci numărul este format din **c** cifre cu valoarea **9**, altfel numărul nostru este format din **c+1** cifre **r**. Nu este necesară generarea şirului şi nici stocarea termenilor.

```
citeşte n
c←[n/9]; r←n%9
┌dacă r=0 atunci
│  ┌pentru i←1,c execută
│  │ scrie 9
│  └■
│altfel
│  ┌pentru i←1,c+1 execută
│  │ scrie r
│  └■
└■
```

100. Numerele generate sunt de forma $\overline{abcd}$ cu **a<b,b>c,c<d** ceea ce determină intervalele de apartenenţă pentru cifre:

a∈[1,8],b∈[a+1,9],c∈[0,b-1],
d∈[c+1,9].

Secvenţa pseudocod afişează numerele cu proprietatea cerută.

```
┌pentru a←1,8 execută
│┌pentru b←a+1,9 execută
│││┌pentru c←0,b-1 execută
│││┌pentru d←c+1,9 execută
│││ scrie a*1000+b*100+c*10+d
│││└■
│└■
└■
```

101. Se prelucrează numerele începând cu ultima cifră. La fiecare pas compară ultima cifră a numărului n cu ultima cifră a numărului m. Se afişează şi se elimină cea mai mare dintre aceste cifre. Algoritmul se încheie atunci când am afişat toate cifrele celor două numere. Eficienţa acestui algoritm este dată de faptul că nu se utilizează variabile suplimentare şi se face o singură parcurgere a şirului de cifre.

```
citeşte n,m
┌repetă
│┌dacă n%10>m%10 atunci scrie n%10; n←[n/10]
│                altfel scrie m%10; m←[m/10]
│└■
└■până când n=0 si m=0
```

102. Se observă că prima grupă din şirul dat conţine un singur element, a doua grupă conţine **1+2** elemente, a treia grupa **2+3** elemente, ..., grupa a **i**-a conţine **(i-1)+i** elemente. Aşadar numărul total de elemente din primele **k** grupe este:

$1+(1+2)+(2+3)+...+((k-1)+k) = (1+2+\ldots+(k-1))+(1+2+\ldots+k) =$

$= k(k-1)/2 + k(k+1)/2 = k^2$.

Vom determina mai întâi grupa din care face parte cel de al n-lea termen, apoi în cadrul grupei, care este acel element. Algoritmul în pseudocod este:

```
citeşte n
k←⌊√n⌋; n←n-k*k; k=k+1
┌dacă n=0 atunci scrie k-1
│ altfel    ┌dacă n<k atunci  scrie n
│           │          altfel scrie k
│           └■
└■
```

103. Cea mai simplă metodă de rezolvare este aceea de a folosi trei contoare, câte unul pentru fiecare dintre cele **3** valori posibile. Aceste contoare vor fi setate pe parcursul citirii celor **n** valori.

```
citeşte n
c1←0; c2←0; c3←0;
┌pentru i←1,n execută
│  citeşte x
│  ┌dacă x=1 atunci c1←c1+1
│  │altfel
│  │  ┌dacă x=2 atunci c2←c2+1
│  │  │         altfel c3←c3+1
└■ └■ └■
┌pentru i←1,c1 executa  scrie 1
└■
┌pentru i←1,c2 executa  scrie 2
└■
┌pentru i←1,c3 executa  scrie 3
└■
```

104. Se observă că numerele din exemplu se formează astfel:

```
1234 = [12345/10]
1235 = [12345/100]*10+12345%10
1245 = [12345/1000]*100+12345%100
1345 = [12345/10000]*1000+12345%1000
2345 = [12345/100000]*10000+12345%10000
```

sau în general

```
nr = [N/(p*10)]*p+N%p
```

Aşadar algoritmul poate fi scris astfel:

```
citeşte N
p←1
┌repetă
│  x←N/(p*10); nr←x*p+N%p; scrie nr; p←p*10
└până când x=0
```

105. Pentru a verifica dacă **a** şi **b** sunt termeni consecutivi în şirul lui Fibonacci o soluţie constă în parcurgerea “drumului invers ” pentru determinarea valorilor din acest şir. Vom porni de la premiza că **a** şi **b** îndeplinesc condiţia din enunţ şi dacă, în urma scăderilor repetate(proces invers celui de adunare, conform formulei de recurenţă din definiţia şirului lui Fibonacci) nu vom ajunge la valoarea primului termen din sir, în cazul nostru **0**, presupunerea noastră iniţială va fi infirmată.

```
citeşte a,b
┌dacă b>a atunci x←a;a←b;b←x
└■
┌cât timp (c>0) execută
│ c←a-b;a←b;b←c
└■
┌dacă c=0
│ atunci scrie "a şi b sunt termeni consecutivi în şirul dat
│ altfel scrie "a şi b NU îndeplinesc condiţia "
└■
```

106. O posibilă variantă de rezolvare este descrisă în algoritmul pseudocod de mai jos:

```
k←0
┌pentru i←10,98,2 execută
│  scrie i
│ ┌dacă i%10=8 atunci salt la rând nou
│ └■
└■
```

107. O posibilă variantă de rezolvare este descrisă în algoritmul pseudocod:

```
k←0
┌pentru i←11,99,2 execută
│  k←k+1; scrie i
│ ┌dacă k%2=0 atunci scrie rând nou
│ └■
└■
```

108. Relaţia $x^2+n=y^2$ o putem prelucra astfel **n=(x-y)*(x+y)** şi putem observa că **x-y** şi **x+y** sunt divizori ai lui **n.** Astfel dacă notăm cu **d** un divizor al lui **n şi** presupunem că $d \le n/d \Rightarrow 1 \le d \le \sqrt{n}$, atunci **x-y=d** şi **x+y=n/d** ⇒ **x=(d+n/d)/2** şi **y=(n/d-d)/2.**

Varianta Pascal	*Varianta C/C++*

`var n,d,k:integer;`
`begin`
`readln(n);`
`k:=0;`
`for d:=1 to trunc(sqrt(n)) do`
` if n mod d=0 then`
`  if not(odd(d+n div d )) then`
`  begin`
`  write((d+n div d)div 2);`
`  write((n div d-d)div 2);`
`  k:=k+1;`
`  end;`
`if k=0 then`
`  write('nu exista');`
`end.`
	`# include <iostream.h>`
	`# include <math.h>`
	`void main()`
	`{`
	`int n,d,k=0;`
	`cin>>n;`
	`for(d=1;d<=sqrt(n);d++)`
	`  if (n%d==0)`
	`  if ((d+n/d)%2==0)`
	`  {k++;`
	`  cout<<(d+n/d)/2;`
	`  cout<<"  "<<(n/d-d)/2`
	`  cout<<endl; }`
	`if (!k) cout<<"\n nu exista ";`
	`}`

109. Se observă din enunţul problemei restricţia pusă asupra lui **n,** astfel putem genera toate numele mai mici sau egale cu **n,** care au cel mult **3 cifre** şi fiecare cifră ia valori din intrervalul **[0,p]**

```
citeşte n,p
┌pentru i←0,p execută
│  ┌pentru j←0,p execută
│  │  ┌pentru k←0,p execută
│  │  │  ┌ dacă (i*100+j*10+k<=n) atunci scrie  i*100+j*10+k
│  │  │  └■
│  │  └■
│  └■
└■
```

110. Vezi rezolvarea problemei 105.

111. Se observă că numărul **x** are cifra de control egală cu cifra de control a numărului **x+9** şi astfel algoritmul este următorul:

```
citeşte a, b; i←a
┌cât timp i≤b execută scrie i; i←i+9
└■
```

112.

```
citeşte n; s←0
┌pentru i←1,n execută citeşte x; s←(s+x)%10
└■
  scrie s
```

113. Vom determina valoarea lui **k** pentru care

$$1+2+3+...+k < n \leq 1+2+...+k+(k+1)$$

adică:

$$k(k+1)/2 < n \leq (k+1)(k+2)/2$$

Astfel termenul al **n**-lea din şir va fi egal cu **k**. Putem determina valoarea lui **k** găsind formula matematică (lăsăm ca exerciţiu această variantă, mai eficientă) sau putem determina această valoare cu ajutorul unui ciclu **cat timp**:

```
citeşte n; k←1; s←0
┌repetă
|  s←s+k; k←k+1
└până când s>=n
scrie k-1
```

114. Se determină mai întâi, în variabila **i**, numărul de cifre ale numărului **n**. Se observă că se vor păstra din n ultimele **i-k** cifre, aşadar numărul solicitat este egal cu $n\%10^{i-k}$. Valoarea lui se calculează folosind în **Pascal** funcţiile **exp** şi **ln**, observând ca $a^b=e^{b*\ln(a)}$. În **C/C++** se va folosi funcţia **pow** din biblioteca **math.h**.

Varianta Pascal	*Varianta C/C++*

`readln(n,k); i:=0; x:=n;`
`while x>0 do`
`  begin  x:=x div 10; i:=i+1; end;`
`if i<=k then begin`
`  x:=trunc(exp((i-k)*ln(10)));`
`  writeln(n mod x);`
` end`
`else  writeln('Numar vid');`
	`cin>>n>>k; i=0; x=n;`
	`while (x>0) { x=x/10; i++;}`
	`if (i<=k) { x=pow(10,i-k);`
	`  cout<< (n%x);`
	`  }`
	`  else cout<<"Numar vid";`

115.

```
citeşte n; max←0
┌pentru i←1,n execută
|  citeste x; k←0; d←2
|  ┌cât timp x>1 execută
|  |  ┌dacă x%d=0 atunci k←k+1
|  |  |                  ┌repetă
|  |  |                  |  x←[x/d]
|  |  |                  └până când x%d<>0
|  |  └■
|  |  d←d+1
|  └■
|  dacă k>max atunci max←k
└■
scrie max
```

116.

```
citeşte n
s←0; d←2
┌cât timp n>1 execută
|  k←0
|  ┌cât timp n%d=0 execută
|  |  k←k+1; n←[n/d]
|  └■
|  s←s+k;  d←d+1
└■
scrie s
```

117.

```
citeşte n; k←0; d←2
┌cât timp n>1 execută
|  ┌dacă n%d=0 atunci
|  |  k←k+1
|  |  ┌repetă
|  |  |  x←[x/d]
|  |  └până când n%d≠0
|  └■
|  d←d+1
└■
scrie k
```

118. Pentru o rezolvare eficienta nu vom reţine numerele în vector, ci vom determina atât valoarea maximă negativă cât şi numărul de apariţii pe măsură ce se citesc numerele. Secvenţa pseudocod ce urmează poate fi o soluţie pentru problema.

```
citeşte n; nrap←0
┌pentru i←1,n execută
|  citeşte x
|   ┌dacă  x<0 atunci
|   |  ┌dacă (nrap=0) sau (x>max) atunci max←x;nrap←1
|   |  |altfel ┌dacă max=x atunci nrap ← nrap+1
|   |  |       └■
|   |  └■
|   └■
└■
┌dacă nrap=0 atunci scrie 'NU EXISTA'
|            altfel scrie max, nrap
└■
```

119. Soluţia propusă va însuma şi număra numerele pare pe măsură ce se citesc.

```
suma ← 0; cate ← 0
citeşte n
┌ dacă n % 2 = 0 atunci  cate ← 1; suma ← n
└■
┌repetă
│    citeşte m
│    ┌dacă n % 2 = 0 atunci cate ← cate + 1; suma ← suma + m
│    └■
└până când n=m
┌dacă cate ≠ 0 atunci scrie suma / cate
│ altfel scrie 'NU EXISTA'
└■
```

120.

```
suma←0; cate←0
┌pentru i←1,10 execută
│   citeşte n
│   ┌dacă n>0 atunci suma←suma+n; cate←cate+1
│   └■
└■
┌dacă cate<>0 atunci scrie  suma/cate cu două zecimale
└■
```

121.

```
citeşte a,b; x←0
┌cât timp (a≠0) and (b≠0) execută
│ cifra←[(a % 10 + b % 10) / 2]
│  x ← x*10 + cifra; a ← [a/10]; b ← [b/10]
└■
c←0
┌cât timp x≠0 execută c ← c*10 + x % 10; x ← [x/10]
└■
scrie c
```

Problema se poate rezolva mult mai simplu, cu 4 instrucţiuni dacă (sau if), dar cu "un mic efort" de memorare, având în vedere că în intervalul **[1..10000]** sunt doar patru numere perfecte (**6, 28, 496, 8128**).

122.

```
citeşte n
┌pentru nr←6,n execută
│  sd←0
│ ┌pentru d=1, [nr/2] execută
│ │   ┌dacă nr % d = 0 atunci
│ │   │    sd←sd+d
│ │   └■
│ └■
│ ┌dacă nr = sd atunci
│ │   scrie nr
│ └■
└■
```

123.

```
   citeşte n; contor ← 0
   ┌pentru i ← 1,n execută
   │  citeşte x; y ← x; sc ← 0
   │ ┌ cât timp y≠0 execută
   │ │  sc ←sc + y%10; y← [y/10]
   │ └■
   │ ┌dacă x % sc =0 atunci
   │ │   contor ← contor + 1
   │ └■
   └■
   scrie contor
```

124.
```
citeşte n; x ← 0; y ← 0
pentru i ← 1,n execută
  citeşte a,b
  dacă a < x atunci  x ← a
  dacă y > b atunci  y ← b
scrie x,y
```

125.
```
citeşte k
pentru cifra ← 1,9 execută
  dacă cifra + k <= 9 atunci
    scrie cifra,cifra+k,cifra,cifra+k,cifra
    dacă cifra + k*2 <= 9 atunci
      scrie cifra,cifra+k,cifra + 2*k,cifra+k,cifra
  dacă cifra - k >= 0 atunci
    scrie cifra,cifra-k,cifra,cifra-k,cifra
    dacă cifra - k*2 >= 0 atunci
      scrie cifra,cifra-k,cifra - 2*k,cifra - k,cifra
```

126. Se verifică dacă cifrele sunt sau nu în ordine decrescătoare (crescătoare începând cu ultima cifră)

```
citeşte n; m ← n; ok ← 0; c←0
 cât timp (m ≠ 0)  şi (ok = 0) execută
   dacă m % 10 < c atunci  ok ← 1
  c ← m % 10;  m ← [m/10]
 dacă ok = 1  atunci scrie 'EXISTA'
  altfel scrie 'NU EXISTA'
```

127. **a)** Numărul de cifre nule aflate pe ultimele poziţii consecutive ale valorii obţinute în urma evaluării lui **n!** reprezintă puterea la care apare **10** în descompunerea lui **n!**. Acest număr este dat de puterile factorilor **2** şi **5** din descompunere şi mai precis de valoarea mai mare. Este suficient deci să calculăm puterea la care apare 5 în descompunerea lui n! . Această putere se determină ca sumă dintre numărul multiplilor lui 5^k mai mici decât **n (k=1,2,...)**.

b) Algoritmul pseudocod reprezintă, conform punctului a), un calcul eficient al numărului cifre nule aflate pe ultimele poziţii consecutive ale valorii obţinute în urma evaluării lui **n!**.

```
citeste n;
f←5; x←0
```

```
┌cât timp f<=n execută
|   x←x+[n/f]
|   f←f*5
└■
scrie x
```

128. **a)** Pentru a determina câte numere naturale mai mici sau egale cu `n` sunt multipli ai lui `a`, dar nu sunt multipli ai lui `b` trebuie să scădem din numărul multiplilor lui a (mai mici sau egali cu n) numărul multiplilor mai mici sau egali cu n ai celui mai mic multiplu comun al celor două numere a şi **b.**

b) Metoda descrisă anterior este reprezentată în pseudocod:

```
citeste n,a,b; c←a ; d←b; r←c%d (restul impartirii)
┌cât timp r>0 execută
| c←d; d←r; r←c%d
└■
cmmmc←a*b/d; scrie [n/a]-[n/cmmmc]
```

129. Secvenţa pseudocod următoare reprezintă o soluţie pentru obţinerea valorii cerute.

```
citeste a,b,c;  r←a%b
┌cat timp r>0 execută  a←b; b←r; r←a%b
└■
 r←c%b
┌cat timp r>0 execută  c←b;b←r;r←c%b
└■
 scrie b (pe ecran)
```

130. Pentru a ne asigura că se afişează cele mai mari **k** numere prime din interval este suficient să parcurgem intervalul **[x;y]** pornind de la **y**.Identificarea unui număr prim determina micşorarea cu o unitate a numărului de numere de afişat, **k.**

Secvenţa pseudocod următoare reprezintă o soluţie pentru problema propusă.

```
citeste x,y,k; aux←k
  ┌cat timp y>=x si k>0 executa
  |    ┌daca prim(y) atunci k←k-1; scrie y
  |    └■
  |  y←y-1
  └■
 ┌daca k>0 atunci
 |  scrie "s-au găsit mai puţine numere prime: ",aux-k
 └■
```

131. Pentru identificarea cifrei de control se observă că suma calculată este de forma **s+c*10**, unde **s** este suma dată de cifrele numărului **n**. Această sumă trebuie să fie divizibilă cu **11**, **(s+c*10)%11=0 → (s+c*11-c)%11=0→ (s-c)%11=0** ceea ce indică faptul că **c=s%11**. Secvenţa pseudocod următoare reprezintă o soluţie pentru identificarea cifrei de control: **citeste n; s←0**

```
┌pentru i←9,1 executa s←s+(n%10)*i; n←[n/10]
└■
c←s%11
┌ daca c=10 atunci  scrie 'x' altfel scrie c
└■
```

132. O metodă eficientă de rezolvare presupunerea generării directe a numerelor din şir. Considerăm **salt** variabila care indică diferenţa dintre doi termeni consecutivi ai şirului. Iniţial saltul este **11**.

Se observă că saltul se modifică atunci când se trece la un alt ordin.

```
citeste n; salt←11; nr←11
┌cât nr<n executa
│       scrie nr
│      ┌daca nr%10=9 atunci nr←salt*10+1;  salt←nr
│      │               altfel nr←nr+salt
│      └■
└■
```

O metodă cel puţin la fel de eficientă de rezolvare, dar care presupune o „doză de curaj" ar fi următoarea: declarăm un vector de constante în care scriem în ordine crescătoare toate numerele care îndeplinesc condiţiile din enunţ (sunt 81 de numere, deci nu ne ia mai mult de 5 minute).Tot ce ne rămâne de făcut în programul principal este să parcurgem acest vector şi să afişăm elementele lui cât timp acestea sunt mai mici decât n.

133. Generăm termeni din şirul Fibonacci până când a=f_n şi b=f_{n+1}. Determinăm coordonatele punctelor, calculăm şi afişăm distanţa dintre cele două puncte.

```
citeste n; a←1 b←1
┌pentru i←3,n executa c←a+b;a←b;b←c
└■
┌daca n=2 atunci b←2
└■
x1←a;y1←b;x2←a+b;y2←x2+y1;
```

scrie $\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$

134. Pentru a evita repetarea unor perechi, alegem a<b. Observăm că fiecare pereche conţine divizori ai numărului **m**, iar cum numerele dintr-o pereche trebuie să fie mai mici decât **n**, **a<=min(n,m/2-1)**. Secvenţa pseudocod următoare reprezintă o soluţie pentru afişarea numerelor cu proprietatea cerută.

```
citeste n; min←n
┌daca [m/2]<n atunci min←[m/2]
└■
┌pentru a←1,min-1 executa
│┌ pentru b←a+1,min executa
│|   ┌daca m%a=0 si n%a=0 atunci
│|   | a_aux←a; b_aux←b
│|   | ┌cat timp b_aux>0 executa
│|   | | r←a_aux%b_aux;a_aux←b_aux;b_aux←r
│|   | └■
│|   | ┌daca a*b/a_aux=m atunci scrie a,b
│|   | └■
│└■ └■
└■
```

135. Fiecare număr din şir este de forma **abc** şi deci singura operaţie pe care trebuie să o efectuăm este inversarea celor două cifre **a** şi **c**.

```
citeste n
┌pentru i←1,n executa
| citeste c; a←c%10; b←[c/10]%10; c←[c/100]+b*10+a*100; scrie c
└■
```

2.1.1.

1. c	2. d	3. a	4. c	5. d	6. b
7. b	8. b	9. c	10. c	11. d	12. c
13. a	14. c	15. b	16. a	17. a	18. c
19. a	20. d	21. b	22. c	23. b	24. b
25. a	26. d	27. a	28. b	29. a	30. d
31. a	32. c	33. c	34. d	35. b	36. c
37. b	38. a	39. b	40. b	41. b	

2.1.2.

1.

1) 1 4

2) 7 (sau 9)

3) Structura

repetă... până când

se poate înlocui cu structura

cât timp...execută.

```
m← [(s+d)/2]; k← k+1
┌dacă c=a_m atunci g←1
│  altfel ┌dacă c<a_m atunci d←m-1
│         │altfel s←m+1
│         └■
└■
┌cât timp not (g=1 sau s>d) execută
│ m← [(s+d)/2];  k← k+1
│┌dacă c=a_m atunci g←1
│└  altfel  ┌dacă c<a_m atunci d←m-1
│           │altfel s←m+1
│           └■
└■
```

2. 1) 12

2) 5 1 3 5 7 9 (sau orice succesiune de **6** valori dintre care prima trebuie să fie **5** iar următoarele fie doar valori impare, fie a doua valoare impară şi restul valorilor mai mici decât a doua, de exemplu: **5 9 2 4 6 8**).

3) Cele două structuri **pentru** se pot înlocui cu două structuri **cât timp...execută:**

```
citeşte n(număr natural nenul)
i←1
┌cat timp i≤n execută
│ citeşte a_i (număr întreg)
│ i ← i+1
└■
m← a_1
i←2
```

```
┌cât timp i≤n execută
│┌dacă a_i este par atunci
│└ ┌dacă m < a_i atunci
│  │  m ← a_i
│  └■
│└■
│i ← i+1
└■
scrie m
```

3.

1) 86522100

2) 4 (cele patru valori sunt **720, 702, 270, 207)**

3) Structura **repetă... până când** se poate înlocui cu structura **cât timp...execută:**

```
i←n%10; a_i←a_i+1; n←[n/10]
┌cât timp n≠0 execută
│  i←n%10; a_i←a_i+1; n←[n/10]
└■
```

4.

1) 1

2) orice secvenţă de **5** numere, în care primul număr este mai mic decât toate celelalte

3) este vorba despre interschimbarea conţinutului variabilelor a_i cu a_{i+1}, în cazul în care ai este mai mic decât a_{i+1}

5.

1) 3

2) 8

4) scrie v(1+n)/2

6.

1) 2, 3, 5, 7, 11, 13, 17, 19

2) 4

3)
```
i← 1
┌cât timp i≤n execută
│  v[i] ← i; i← i+1
└■
```

7.

1) 6,9,12,21 şi 4

2) De exemplu, **14** (un număr care să nu dividă nici un element al vectorului)

3) Pentru a fi afişate elementele vectorului trebuie să-l aibă pe **x** ca divizor.

8.

1) 4 0 1 6 3 10 12 4 16

2) 1 2 3 4 5

4) a[i] % 2 ≠ 0

9. **1)** 77 5

2) Orice secvenţă formată din **7** numere divizibile cu **7**.

3) Orice secvenţă formată din **5** numere nedivizibile cu **7**.

10. **1)** 0 1 6 7 8

2) Orice secvenţă formată din **7** numere întregi ordonate crescător

4) Structura **cât timp...execută** poate fi înlocuită cu structura **repetă...cât timp.**

```
┌dacă j>0 şi x<a_j atunci
│┌repetă
│| a_{j+1}←a_j; a_j←x; j←j-1
│└■cât timp j>0 şi x<a_j
└■
```

11. Dacă numărul **n** nu are toate cifrele în ordine descrescătoare (crescătoare începând cu ultima cifră) putem obţine un număr mai mare decât acesta inversând cele două cifre care nu respectă această proprietate. Cifrele numărului le memorăm în vectorul v începând cu cifra unităţilor şi pe măsură ce obţinem cifrele se şi inversează pentru a obţine numărul mai mare. În final se construieşte valoarea noului număr.

```
citeşte n
m ← n; v_1 ← m%10; m ← [m/10]; i ← 1
┌ cât timp m≠0 execută
│  i ← i+1; v_i ← m%10
│  ┌dacă v_i < v_{i-1} atunci  aux ← v_i , v_i ← v_{i-1} , v_{i-1} ← aux
│  └■
│  m ← [m/10]
└■
```

```
┌ cât timp i≠0 execută m ← m * 10 + v_i; i ← i-1
└■
┌dacă m > n atunci scrie m
│ altfel scrie 'NU EXISTA'
└■
```

12. Secvenţa program corespunzătoare rezolvării cerinţei problemei:

Varianta Pascal	*Varianta C/C++*

`.....`
`n:=0;`
`for i:=8 downto 2 do`
`  for j:=i-2 downto 0 do`
`   begin inc(n);a[n]:=10*i+j; end;`
`.....`
	`.....`
	`for(n=0,i=8; i>=2; i-=2)`
	`   for(j=i-2; j>=0; j-=2)`
	`      a[++n]=10*i+j;`
	`.....`

Deoarece sunt puţine numere care îndeplinesc condiţiile din enunţ (10 numere), se poate da şi o soluţie în care se construieşte vectorul sub forma unei constante cu tip (Pascal) sau se atribuie valori vectorului prin declaraţie (C/C++)

13. S-a notat cu **[z]** partea întreagă a numărului real **z.** Secvenţa pseudocod următoare reprezintă o soluţie pentru scrierea programului:

```
┌pentru i=1,15 execută
│  x←[a_i]; y←x+1
│  ┌dacă x=a_i atunci y←y-1
│  └■
│  scrie y,x
└■
```

14. Vom citi cele două secvenţe în doi vectori diferiţi şi apoi, folosind un algoritm similar algoritmului de interclasare vom determina valorile comune celor doi vectori. Acest algoritm îl schiţăm mai jos.

```
citeşte m,n
┌pentru i←1,m execută citeşte a[i]
└■
┌pentru i←1,n execută citeşte b[i]
└■
i←1; j←1
┌cât timp i<=m şi j<=n execută
│ ┌dacă a[i]=b[j] atunci scrie a[i]; i←i+1; j←j+1
│ │  altfel ┌dacă a[i]<b[j] atunci i←i+1
│ │         │               altfel j←j+1
│ │         └■
│ └■
└■
```

Problema s-ar putea rezolva şi cu un singur vector în care se reţin primele m valori citite; celelalte n valori se citesc pe măsură ce se aplică algoritmul de interclasare.

15. Se memorează într-un vector culorile celor **n** ţări, apoi se verifică dacă două ţări vecine au aceeaşi culoare. În algoritmul de mai jos, **i div 2** reprezintă câtul împărţirii lui **i** la **2**.

```
citeşte n
┌pentru i←1,n execută citeşte cul[i]
└■
i←2; x←1
┌cât timp i≤n şi x=1 execută
│   ┌dacă cul[i]=cul[i div 2] atunci x←0
│   └■
│  i←i+1
└■
┌dacă x=1 atunci scrie ”corect”
│         altfel scrie ”incorect”
└■
```

16. La citirea numerelor din şir se vor reţine într-un vector doar valorile impare, şi apoi se va sorta şi afişa acest vector.

```
n←0; citeşte x
┌cat timp x>=0 executa
│   ┌daca x%2=1 atunci n←n+1; a[n]=x
│   └■
│  citeste x
└■
┌pentru i←1,n-1 executa
│   ┌pentru j←i+1,n executa
│   │   ┌daca a[i]>a[j] atunci x←a[i]; a[i]←a[j]; a[j]←x
│   │   └■
│   └■
└■
┌pentru i←1,n executa scrie a[i]
└■
```

17. Se citesc numerele şi se reţin într-un vector doar cele care îndeplinesc condiţia din enunţ. Pentru a afla prima cifra a unui număr vom împărţii numărul la **10** atâta timp cât el este mai mare decât **9**.

```
citeşte n; m←0
┌pentru i←1,n executa
│  citeste x; y←x
│   ┌cat timp y>9 executa y←[y/10]
│   └■
│   ┌daca y=x%10 atunci m←m+1; a[m]←x
│   └■
└■
┌pentru i←1,m-1 execută
│   ┌pentru j←i+1,m execută
│   │   ┌dacă a[i]>a[j] atunci x←a[i]; a[i]←a[j]; a[j]←x
│   │   └■
│   └■
└■
┌pentru i←1,m execută scrie a[i]
└■
```

18. Numerele se citesc într-un vector a, se ordonează descrescător şi se calculează suma primelor **n/2** elemente. Deoarece enunţul nu cere o soluţie eficientă, se poate utiliza o metodă oarecare pentru sortarea valorilor din vector.

```
citeste n
pentru i=1,n execută citeste a[i]
■
m←n
repeta
  ordonat←1
  pentru i=1,m-1 execută
    daca a[i]<a[i+1] atunci
      aux←a[i];a[i]←a[i+1];a[i+1]←aux; ordonat←0
    ■
  ■
m←m-1
pana cand ordonat=1
s←0
pentru i←1,n/2 executa s←s+a[i]
■
scrie s
```

19. Primele **[n/2]** poziţii conţin numere impare, fiecare element de pe poziţia **i, i∈[1,[n/2]],** având valoarea **2*i-1** iar elementul simetric valoarea **2*i**.Dacă numărul de elemente este impar elementul din mijloc are valoarea n.

```
pentru i←1,[n/2] execută
  a[i]←2*i-1; a[n+1-i]←2*i
■
dacă n%2=1 atunci a[(n+1)/2]←n
■
```

O altă metodă de completare a vectorului atribuie mai întâi primului element valoarea **1** şi ultimului element valoarea **2.** Celelalte elemente se completează crescător, fiecare element **a[i]←a[i-1]+2, i∈[2,[(n+1)/2]],** respectiv:

a[i]=a[i+1]+2, i∈[(n+1)/2+1,n]

```
a[1]←1;a[n]←2
pentru i←2,[(n+1)/2] execută
  a[i]←a[i-1]+2
■
pentru i←n-1,[(n+1)/2]+1,-1 execută
  a[i]←a[i+1]+2
■
```

2.2.1.

1. b	2. a	3. c	4. a	5. c	6. d
7. a	8. a	9. c	10. d	11. c	12. b
13. c	14. b	15. b	16. d	17. b	18. a
19. a					

2.2.2.

1. Secvenţa pseudocod următoare reprezintă o soluţie pentru construirea matricei:

```
pentru i=1,n execută
  pentru j=1,n execută
    dacă (i≠j) and (i+j≠n+1) atunci a_ij←2
                             altfel a_ij←1
  ■
■
```

2. Următoarea secvenţă pseudocod reprezintă o soluţie pentru construirea matricei:

```
pentru i=1,n execută
  pentru j=1,n execută
    dacă (j>i)and(j<n+1-i)or(j>n+1-i)and(j<i)
      atunci a_ij←1
      altfel  dacă (i=j)or(i+j=n+1) atunci a_ij←0
                                    altfel a_ij←2
```

3. Următoarea secvenţă pseudocod reprezintă o soluţie pentru construirea matricei:

```
pentru i=1,n execută
  pentru j=1,n execută
    dacă (j>=i)and(j<=n+1-i)or(j>=n+1-i)and(j<=i)
      atunci a_ij←1
      altfel a_ij←2
```

4. Secvenţa pseudocod următoare reprezintă o soluţie posibilă pentru construirea matricei:

```
pentru i=1,n execută
  pentru j=1,n execută
    dacă i=[n/2]+1 or j=[n/2]+1 atunci a_ij←1
       altfel a_ij←3
a_[n/2]+1, [n/2]+1←2
```

5. Secvenţa pseudocod următoare reprezintă o soluţie posibilă pentru construirea matricei:

```
pentru i=1,n execută
 k←i
 pentru j=1,n execută
   a_i,j←k
   dacă j<n+1-i atunci k←k+1
                altfel k←k-1
```

6. O rezolvare posibilă: presupunem că numerotăm liniile şi coloanele matricei de la **1** la **n**. Atribuim fiecărui element al matricei valorea unei variabile **k** iniţializată cu **1** şi a cărei valoare o mărim cu **1** după fiecare atribuire; parcurgem liniile matricei cu număr de ordine impar de la stânga la dreapta (crescător) şi celelalte de la dreapta la stânga (descrescător); secvenţa pseudocod corespunzătoare este următoarea:

```
k←1
pentru i←1,n execută
  dacă i%2≠0 atunci
    pentru j←1,n execută a_i,j← k; k←k+1
  altfel  pentru j←n,1,-1 execută a_i,j← k; k←k+1
```

7. S-a notat cu **x%y** restul împărţirii numărului natural **x** la numărul natural **y**. Secvenţa pseudocod următoare reprezintă o soluţie pentru construirea matricei (în forma din exemplul dat în enunţ):

```
pentru i=1,n execută
  pentru j=1,n execută a_ij←(i+j)% n + 1
```

8. Secvenţa pseudocod următoare reprezintă o soluţie pentru construirea matricei:

```
k←0; p←0; citeste n
pentru i=1,n execută
  a_ii←0
  pentru j=1,i-1 execută k←k+1; a_ij←k
  pentru j=i+1,n execută p←p+1; a_ij←p
```

9. S-a notat cu **x%y** restul împărţirii numărului natural **x** la numărul natural **y**. Secvenţa pseudocod următoare reprezintă o soluţie pentru construirea matricei:

```
pentru p=1,[(n+1)/2] execută
  x←p%2
  pentru i=p, n+1-p execută
      a_ip←x; a_pi←x; a_i,n+1-p←x; a_n+1-p,i←x
```

10. Secvenţa pseudocod următoare reprezintă o soluţie pentru construirea matricei:

```
pentru i=1,n execută
  pentru j=1,n execută
    dacă i+j=n+1 atunci a_ij←n+1
                 altfel a_ij←i
```

11. Secvenţa pseudocod următoare reprezintă o soluţie pentru construirea matricei:

```
k←1
pentru i=1,n execută
  dacă i%2 ≠ 0
  atunci pentru j=1,n execută a_ij←k; k←k+1
  altfel pentru j=n,1,-1 execută a_ij←k; k←k+1
```

S-a notat cu **x%y** restul împărţirii numărului întreg **x** la numărul întreg **y**.

12. Secvenţa pseudocod următoare reprezintă o soluţie pentru construirea matricei:

```
u←m%10; m←m/10; z←m%10
m←m/10; s←m%10; m←m/10
pentru i=1,n execută
  pentru j=1,2*n-1 execută
    dacă i=n or j=n atunci a_i,j←0; a_2*n-i,2*n-j←0
    altfel  dacă j<n atunci a_i,j←u; a_2*n-i,2*n-j←m
                       altfel a_i,j←z; a_2*n-i,2*n-j←s
```

S-a notat cu **x%y** restul împărţirii numărului întreg **x** la numărul întreg **y.**

13. Algoritmul pseudocod de rezolvare a acestei probleme este următorul:

```
citeşte m,n; s←0
┌pentru i←1,m execută
│  citeşte a;  max←a
│ ┌pentru j←2,n execută
│ │ citeşte a
│ │ ┌dacă a>max atunci max←a
│ │ └■
│ └■
│ s←s+max
└■
scrie s
```

14. Algoritmul este unul foarte simplu de lucru cu matrice. Vom completa întreaga matrice printr-o singură parcurgere a ei. Prezentăm mai jos algoritmul pseudocod corespunzător:

```
citeşte n
┌pentru i←1,n execută
│ ┌pentru j←1,n execută
│ │  ┌dacă i=j atunci  a_{i,j}=0
│ │  │altfel  ┌dacă i<j atunci a_{i,j}=i
│ │  │        │  altfel a_{i,j}←j
│ │  │        └■
│ │  └■
│ └■
└■
```

15. Pentru sortarea liniilor matricii se poate folosi orice metodă cunoscută. Aici vă propunem metoda bulelor. Algoritmul în pseudocod este următorul:

```
citeste n
┌pentru i←1,n execută
│ ┌pentru j←1,m executa citeste a_{ij}
│ └■
└■

┌pentru i←1,n execută
│  ┌repeta
│  │  ok←1
│  │  ┌pentru j←1,m-1 executa
│  │  │  ┌daca a_{ij}<a_{i,j+1} atunci t←a_{ij}; a_{ij}←a_{i,j+1}; a_{i,j+1}←t; ok←0
│  │  │  └■
│  │  └■
│  └pana cand ok=1
└■  scrie matricea
```

16. Elementele de pe diagonala principală sunt cele de forma $a_{i,i}$. Secvenţa de sortare, folosind metoda bulelor (se poate folosi orice alta metodă cunoscută), arată în pseudocod astfel:

```
repetă
  ok←1
  pentru i←1,n-1 execută
    dacă a[i,i]<a[i+1,i+1] atunci
      t←a[i,i]; a[i,i]←a[i+1,i+1]; a[i+1,i+1]←t
      ok←0
  sfdacă
sfpentru
pana cand ok=1
```

17. Pentru a determina suma cerută nu este necesar să memorăm matricea. Vom citi elementele matricii, iar dacă elementul se află pe rama matricii şi este număr par îl vom contoriza. Algoritmul în pseudocod arată astfel:

```
citeste n
k←0
pentru i←1,n executa
  pentru j←1,n executa
    citeste x
    daca x%2=0 si ((i=1) sau (j=1) sau (i=n) sau (j=n))
      atunci k←k+1
  sfdaca
sfpentru
scrie k
```

18. Se pot aborda diverse strategii de completare a elementelor matricei (de exemplu prin backtracking). O strategie eficientă însă trebuie să urmărească completarea directă a liniilor cu valori care respectă cerinţele din enunţ (fără să fie nevoie de verificarea îndeplinirii acestor proprietăţi). O posibilitate ar fi următoarea:

- primele **n** linii se completează ca la matricea unitate;
- în următoarele **n-2** linii vor fi trei elemente egale cu **1**: ultimul şi un grup de două elemente cu poziţii alăturate, acest grup fiind „deplasat" de la o linie la alta cu câte o coloană. Avem posibilitatea să completăm **n-2** linii în acest mod deoarece primul **1** din cele trei „pleacă" din coloana **1** şi „termină" in coloana **n-2**.

```
1 0 0 0 0   - cinci linii
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1
1 1 0 0 1   - trei linii
0 1 1 0 1
0 0 1 1 1
   total    = opt linii
```

În exemplul alăturat, această strategie este aplicată pentru **n=5**:

19. Deoarece majoritatea elementelor matricei sunt nule, ar fi ineficientă memorarea acesteia, motiv pentru care observăm că pe fiecare linie a matricei sunt exact **3** valori nenule. Înaintea elementului de valoare **k,** în matrice sunt completate **k/3** linii, deci acest element se va afla pe linia următoare şi anume pe linia **k/3+1.**

Pentru a determina coloana pe care se află valoarea **k,** vom observa că acestă valoare este precedată de **linie-2** elemente nule(linie fiind linia pe care se află pozitionat elementul **k**) şi **k%3** elemente nenule, astfel coloana va fi **linie-2+k%3+1= k%3+k/3**

Varianta Pascal	*Varianta C/C++*
`var k:integer;`	`# include <iostream.h>`
`begin`	`void main()`
`readln(k);`	`{int k;  cin>>k;`
`writeln('linia=',k div 3+1);`	`  cout<<"linia = "<<k/3+1;`
`write('coloana=',k mod 3+k div 3);`	`  cout<<"coloana = "<<k%3+k/3;`
`end.`	`}`

20. Pentru a contoriza coloanele cu elemente distincte, vom parcurge în ordine coloanele matricei şi pentru fiecare coloană, un indicator va marca proprietatea enunţată(de a avea sau nu elemente distincte).

Pentru a verifica această proprietate fiecare element al coloanei curente va fi comparat cu toate elementele aflate pe liniile următoare şi pe aceeaşi coloană cu el.

```
c←0
┌pentru j←1,m execută
│  distincte←1
│  ┌pentru i←1,n-1 execută
│  │  ┌pentru k←i+1,n execută
│  │  │  ┌dacă aij=akj atunci  distincte←0
│  │  │  └■
│  │  └■
│  └■
│  ┌dacă distincte=1 atunci  c←c+1
│  └■
└■
```

21.

```
  citeşte n
┌pentru i←1,n*(n-1)/2 execută
│   citeşte vi
└■
k←1
┌pentru i←1,n-1 execută
│  ┌pentru j←i+1,n execută aij←vk; aji←vk;k←k+1
│  └■
└■
```

22.

```
dim←0
┌pentru i←1,m execută
│  ┌pentru j←1,n execută
│  │  Vdim←aij
│  │  dim ← dim + 1
│  └■
└■
e_palindrom←1
┌pentru i←1, n*m/2 execută
│  ┌dacă vi<>vn*m-i+1 atunci  e_palindrom←0
│  └■
└■
┌dacă e_palindrom=0 atunci scrie ”NU” altfel scrie ”DA”
└■
```

23.
```
citeşte n,m
┌pentru i←1,n execută
│  ┌pentru j←1,m execută citeşte a[i,j]
│  └■
└■
citeşte k; max←a[k,1]
┌pentru j←2,m execută
│  ┌dacă a[k,j]>max atunci max←a[k,j]
│  └■
└■
afişează max
```

24.
```
citeşte m,n
nr←1
┌pentru i←1,m execută
│  ┌pentru j←1,n execută  a[i,j]←nr; nr←nr+2
│  └■
└■
afişare matricea a
```

25.
```
citeşte m,n
┌pentru i←1,m execută
│  ┌pentru j←1,n execută
│  │  ┌dacă i>j atunci a[i,j]←j*10+i
│  │  │         altfel a[i,j]←i*10+j
│  │  └■
│  └■
└■
afişare matricea a
```

26. Observăm că elementele egale se găsesc pe aceeaşi paralelă la diagonala secundară

```
┌pentru i=1,n execută
│┌pentru j=1,n execută a[i,j]←(i+j-2)%n+1
│└■
└■
```

27. Vom parcurge matricea pe coloane şi vom utiliza un contor pentru a genera valorile de la **1** la n^2

```
contor←1
┌pentru j←1,n execută
│┌pentru i←1,n execută
│|  a[i,j]←contor; contor←contor+1
│└■
└■
```

28. Deoarece pe fiecare linie există un singur elemet situat pe diagonala principală si unul pe cea secundară, vom calcula direct diferenţa adunând elementul de pe diagonala principală şi scăzându-l pe cel de pe diagonala secundară.

```
s←0
┌pentru i←1,n execută s←s+a[i,i] - a[i,n+1-j]
└■
scrie s
```

29.
```
max=0
pentru i←1,n execută
 s←0
 pentru j←1,n execută s←s+a[i,j]
 ■
 dacă s >max atunci max←s; imax←i
 ■
■
scrie imax
```

30. Vom borda matricea cu elemente 0 asfel încât şi elementelor de pe marginile matricei să le calculăm suma vecinilor cu aceeaşi formulă.

```
pentru i←1,n execută
   a[i,0] ← 0; a[i,n+1] ← 0; a[0,i] ← 0; a[n+1,i] ← 0
■
pentru i←1,n execută
   pentru j←1,n execută
      s←a[i-1,j-1] +a[i-1,j] +a[i-1,j+1]+a[i,j-1]+a[i,j+1]+a[i+1,j-1]+a[i+1,j]+a[i+1,j+1]
      dacă s % a[i,j]=0 atunci scrie a[i,j]
      ■
   ■
■
```

31. Următoarea secvenţă pseudocod reprezintă o soluţie pentru construirea matricei:

```
pentru i←1,n execută
 a[ii]←1
 pentru j←i+1,n execută a[ij]←0; a[ji]←2
 ■
■
```

32. O matrice care respectă condiţiile din enunţ este cea obţinută prin generarea permutărilor circulare ale mulţimii **{1,2,…,n}.**

De exemplu, pentru **n=3** se generează matricea alăturată.

```
1 2 3
2 3 1
3 1 2
```

Următoarea secvenţă pseudocod reprezintă o soluţie pentru construirea matricei:

```
pentru i←1,n execută
 pentru j←1,n execută a[ij]←(i+j)% n+1
 ■
■
```

33. Următoarea secvenţă pseudocod reprezintă o soluţie pentru construirea matricei:

```
pentru i←1,n execută
 pentru j←1,n execută a[ij]←i*10+j
 ■
■
```

2.3.1.

1. d	2. c	3. a	4. d	5. c	6. c
7. a	8. c	9. a	10. b	11. a	12. a
13. d	14. d	15. c	16. a	17. d	18. a
19. c	20. a	21. c	22. a	23. b	

2.3.2.

1. *Varianta Pascal*

```
var s:string[20];
    i,t:byte;
begin  readln(s); t:=0;
 for i:=1 to length(s) do
 if pos(s[i],'aeiouy')>0
    then inc(t);
 if 2*t=length(s)
   then writeln('DA')
   else writeln('NU');
end.
```

Varianta C++

```
#include <string.h>
#include <iostream.h>
char s[21];  int i, t;
void main()
{cin.get(s,21); t=0;
 for (i=0;i<strlen(s);i++)
   if (strchr("aeiouy",s[i])) t++;
 if (2*t==strlen(s)) cout<<"DA";
   else cout<<"NU";}
```

2. Se parcurge primul şir şi se verifică dacă fiecare caracter al acestuia se regăseşte în al doilea şir.

Varianta Pascal

```
var s1,s2:string[100];
    i,ok:byte;
begin  ok := 1;
 readln(s1); readln(s2);
 for i:=1 to length(s1) do
  if pos(s1[i],s2)=0
    then ok:=0;
 if ok=1 then
   then writeln('DA')
   else writeln('NU');
end.
```

Varianta C++

```
#include<string.h>
#include<iostream.h>
char s1[101], s2[101];
int i,ok=1;
void main()
{cin>>s1; cin>>s2;
 for (i=0;i<strlen(s1);i++)
   if(strchr(s2,s1[i])==NULL)
      ok=0;
 if (ok==1) cout<<"DA";
   else cout<<"NU";}
```

3. *Varianta Pascal*

```
var n:longint;
    s:string[10];
    c:char;
    k:integer;
begin
   readln(n);  str(n,s);
   k:=length(s);
   c:=s[1]; s[1]:=s[k];
   s[k]:=c; val(s,n,k);
writeln(n,' ',sqrt(n):1:2)
end.
```

Varianta C/C++

```
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <iostream.h>
#include <iomanip.h>
void main()
{ char s[10]; long n;
  cin>>n;  ltoa(n,s,10);
  char c; int k=strlen(s)-1;
  c=s[0]; s[0]=s[k]; s[k]=c;  n=atol(s);
  cout<<n<<' '<<setprecision(2)<<sqrt(n);
}
```

4. Pentru a verifica dacă cele două numere au cifre comune putem căuta, pe rând, cifrele primului număr în cel de-al doilea număr sau putem utiliza doi vectori cu câte 10 elemente, câte unul pentru fiecare număr, vectori care păstrează pe fiecare poziţie **c**, **c∈[0,9]**, numărul de apariţii ale cifrei c în numărul corespunzător. O poziţie pentru care elementele din ambii vectori sunt nenule indică prezenţa acelei cifre în cele două numere.

Metoda de rezolvare prezentată în continuare presupune transformarea numerelor în şiruri de caractere şi căutare primului caracter cifră comun.

Varianta Pascal

```
var a,b:longint; sa,sb:string[11];
    k:boolean; c:char;
begin
readln(a,b); k:=false;c:='0';
while not k and c<='9' do begin
if(pos(c,sa)>0) and (pos(c,sb)>0)
   then k:=false;
c:=succ(c)
end;
if k then write('DA')
     else write('NU')
end.
```

Varianta C/C++

```
# include <iostream.h>
# include <string.h>
# include <stdlib.h>
long a,b;
char sa[11],sb[11],c; int k;
void main()
{cin>>n>>m;k=1;
 ltoa(a,sa,10);ltoa(b,sb,10);
 for(c='0';c<='9' && !k;c++)
 if(strchr(sa,c) && strchr(sb,c))k=1;
 if(k)cout<<"DA";
      else cout<<"NU";}
```

5. *Varianta Pascal*

```
var s,t:string[81];
    i,k:byte;
begin readln(t);s:=t; t:='';
for i:=1 to length(s) do
begin
  inc(k); t:=t+s[i];
  if s[i]=' ' then
   begin
     while s[i]=' ' do inc(i);
     dec(i);
   end;
end;
writeln(t)
end.
```

Varianta C/C++

```
#include <string.h>
#include <iostream.h>
void main()
{ char t[81],sep[]=" ";
  char s[81] ="",,*p;
  cin.get(t,81); p=strtok(t,sep);
  if(p)
  { strcat(s,p);
    do
    { p=strtok(NULL,sep);
     if(p)
     { strcat(s,sep);strcat(s,p);}
    }while(p);
   }
  strcpy(t,s);cout<<t;}
```

6. Se parcurg simultan ambele şiruri, până la epuizarea unuia dintre ele, apoi se copiază restul caracterelor din şirul neepuizat. Secvenţele de program de mai jos trebuie să fie completate cu declaraţiile de variabile, includerea bibliotecilor necesare etc.

Varianta Pascal

```
...
readln(s1); readln(s2);
m:=length(s1); n:=length(s2);
i:=1;
while (i<=m) and (i<=n) do
begin
  s3:=s3+s1[i]+s2[i]; i:=i+1;
end;
if m<n then
  s3:=s3+copy(s2,m+1,n)
else
  s3:=s3+copy(s1,n+1,m);
writeln(s3);
```

Varianta C/C++

```
...
cin>>s1>>s2;
m=strlen(s1); n=strlen(s2);
i=0; k=0;
while ((i<m)&&(i<n))
{  s3[k++]=s1[i];
   s3[k++]=s2[i]; i++;
}
if (m<n) strcpy(s3+k,s2+m);
else strcpy(s3+k,s1+n);
s3[m+n]='\0';
cout<<s3;
```

7. Vom folosi un algoritm de interclasare, afişând şirul final direct pe ercan, fără să-l construim efectiv în memorie. Eficienţa acestei rezolvări este dată de eficienţa algoritmului de interclasare..Secvenţa de program care realizează interclasarea este următoarea:

Varianta Pascal	*Varianta C/C++*

`n:=length(s1); m:=length(s2);`
`i:=1; j:=1;`
`while (i<=n) and (j<=m) do`
` if s1[i]<=s2[j] then`
`  begin write(s1[i]); i:=i+1; end`
` else`
`  begin write(s2[j]); j:=j+1; end;`
`while i<=n do`
`  begin write(s1[i]); i:=i+1; end;`
`while j<=m do`
` begin write(s2[j]); j:=j+1; end;`
	`n=strlen(s1); n=strlen(s2);`
	`i=j=0;`
	`while ((i<n) && (j<m))`
	`  if (s1[i]<s2[j])`
	`  {  cout<<s1[i]; i++;}`
	`  else`
	`  {  cout<<s2[j]; j++;}`
	`while (i<n)`
	`{  cout<<s1[i]; i++;}`
	`while (j<m)`
	`{  cout<<s2[j]; j++;}`

8. In limbajul **Pascal**, inserarea unei litere în şirul dat se face cu ajutorul procedurii de bibliotecă **insert**. În limbajul C/C++, pentru inserarea unei litere vom muta spre dreapta cu o poziţie toate literele începând cu poziţia de inserare.

Varianta Pascal	*Varianta C/C++*

`i:=1`
`while i<=length(s) do`
` begin`
`  case s[i] of`
`   'a': insert('b',s,i+1);`
`   'e': insert('f',s,i+1);`
`   'i': insert('j',s,i+1);`
`   'o': insert('p',s,i+1);`
`   'u': insert('v',s,i+1);`
`  end;`
`  i:=i+1;`
` end;`
`s:=s+'stop';`
`writeln(s);`
	`char *s,*t; int i;`
	`gets(s);`
	`for(i=0;i<strlen(s);i++)`
	` if (strchr("aeiou",s[i]))`
	` {  strcpy(t,s+i+1);`
	`  strcpy(s+i+2,t);`
	`  switch(s[i])`
	`  {case 'a': s[i+1]='b'; i++; break;`
	`  case 'e': s[i+1]='f'; i++; break;`
	`  case 'i': s[i+1]='j'; i++; break;`
	`  case 'o': s[i+1]='p'; i++; break;`
	`  case 'u': s[i+1]='v'; i++; break;`
	`  } }`
	`strcat(s,"stop");`
	`cout<<s;`

O soluţie mai simplă ar fi să nu construim efectiv în memorie şirul cerut de enunţ ci doar să-l afişăm pe ecran. În acest caz, nu mai e nevoie de inserări.

9. Parcurgem şirul caracter cu caracter, şi numărăm vocalele în variabila **k**. În variabila **j** vom reţine poziţia primului caracter din cuvântul curent. Iniţial **j=0**. În momentul întâlnirii unui spaţiu, înseamnă că s-a terminat un cuvânt, deci dacă **k<=3**, vom afişa cuvântul curent (care începe din poziţia **j** până la poziţia **i-1**). Reiniţializăm contorul **k** la **0**, iar **j** va primi ca valoare poziţia unde începe următorul cuvânt (**i+1**).

Varianta Pascal	*Varianta C/C++*

`readln(s);`
`if (s[length(s)]<>' ') then`
`  s:=s+' ';`
`i:=0;`
`j:=0;`
`k:=0;`
	`gets(s);`
	`if (s[strlen(s)-1]!=' ')`
	`  strcat(s," ");`
	`i=0;`
	`j=0;`
	`k=0;`

```
while i<length(s) do
begin
 if s[i]=' ' then begin
   if k<=3 then
     writeln(copy(s,j,i-j));
   j:=i+1; k:=0;
  end
 else
  if pos(s[i],'aeiou')>0 then
    k:=k+1;
 i:=i+1; end;
```

```
while (i<strlen(s))
{if (s[i]==' ')
 { if (k<=3)
   { strncpy(t,s+j,i-j);
     t[i-j]='\0';
     cout<<t<<'\n';}
   j=i+1; k=0;
 }
 else
 if(strchr("aeiou",s[i])) k++;
 i++; }
```

10. Vom parcurge şirul caracter cu caracter şi la întâlnirea unei vocale, o ştergem din şir şi afişăm şirul obţinut.

Varianta Pascal

```
readln(s); i:=1;
while i<=length(s) do begin
 if pos(s[i],'aeiou')>0 then
  begin
   delete(s,i,1);
   writeln(s);
  end
 else i:=i+1;
end
```

Varianta C/C++

```
gets(s);
i=0;
while (i<strlen(s))
{if(strchr("aeiou",s[i]))
 { strcpy(s+i,s+i+1);
   cout<<s<<'\n';
 }
 else i++;
}
```

11. O posibilă soluţie este următoarea:

Varianta Pascal

```
voc := 'aeiou';
readln(s);
for i := 1 to length(voc) do
begin k :=0; p := pos(voc[i],s);
    while p<>0 do
    begin delete(s,p,1);inc(k);
          p := pos(voc[i],s);
    end;
    if k>0 then writeln(s)
end
```

Varianta C/C++

```
char voc[]="aeiou";
gets(s);
for (i=0;i<strlen(voc);i++)
{ k=0;
  while (p=strchr(s,voc[i]))
  { strcpy(p,p+1);
    k++;
  }
  if (k>0) cout<<s<<endl;
}
```

12. O soluţie simplă pe care o vom descrie în cele ce urmează constă în parcurgerea caracter cu caracter a cuvântului **cuv1** şi memorarea de **i** ori a celui de-al **i**-lea caracter, în noul cuvânt format **cuv2**.

Varianta Pascal

```
for i:=1 to length(cuv1) do
  for j:=1 to i do
    cuv2:=cuv2+cuv1[i];
```

Varianta C/C++

```
k=0;
 for(i=0;i<strlen(cuv1);i++)
    for(j=0;j<=i;j++)
      cuv2[k++]=cuv1[i];
cuv2[k]='\0';
```

13. ***Varianta Pascal***

```
var s:string[20];
    n:integer;
begin readln(s);
n:=length(s);
```

Varianta C/C++

```
void main()
{char s[21];
 int n;
 cin>>s;
```

```
if(n mod 2 = 0) then
     delete(s,n div 2,2)
else delete(s,n div 2+1,1);
if length(s)=0 then
     writeln('SIR VID')
else writeln(s)
end.
```

```
 n=strlen(s);
 if(n%2==0)
    strcpy(s+n/2-1,s+n/2+1);
 else strcpy(s+n/2,s+n/2+1);
 if(strlen(s)!=0) cout<<s;
 else  cout<<"SIR VID";
}
```

14. *Varianta Pascal*

```
var fraza,fmax:string;
    i,j,n,max,nrc:integer;
begin readln(n);
  for i:=1 to n do begin
     readln(fraza);
     nrc:=0;
     for j:=1 to length(fraza) do
        if fraza[j]=' ' then
            inc(nrc);
     if nrc>max then begin
         max:=nrc;
         fmax:=fraza
     end
  end;
  writeln(fmax);
end.
```

Varianta C/C++

```
void main()
{char fraza[256],fmax[256];
 int n,i,max=0,nrc,j;
 cin>>n;
 for(i=0;i<n;i++)
  {gets(fraza); nrc=0;
   for(j=0;j<strlen(fraza);j++)
      if(fraza[j]==' ')nrc++;
   if(nrc>max)
      {max=nrc;
       strcpy(fmax,fraza);
       }
   }
 cout<<fmax;
}
```

15. Numărul minim de cuvinte din şir este egal cu numărul poziţiilor pe care se găsesc litere diferite în cele două cuvinte+1. Generarea şirului începe cu primul cuvânt şi se analizează pe rând fiecare literă. Pentru orice poziţie **i** pentru care **u[i] ≠ v[i]** se generează un nou cuvânt **u**, prin atribuirea **u[i]=v[i]**.

Varianta Pascal

```
var u,v:string[100];
    i:integer;
begin readln(u,v);
if length(u)=length(v) do
begin
 writeln(u);
 i:=1;
 while u<>v do begin
  if u[i]<>v[i] do begin
      u[i]:=v[i];
      writeln(u)
     end;
  i:=i+1
 end
else write('EROARE')
end.
```

Varianta C/C++

```
# include <iostream.h>
# include <string.h>
char u[101],v[101]; int i;
void main()
{ cin>>u>>v;
if(strlen(u)==strlen(v))
{ i=0; cout<<u<<'\n';
  while(strcmp(u,v))
    {if(u[i]!=v[i])
     {u[i]=v[i];
      cout<<u<<'\n';}
      i++;}
 else cout<<" EROARE";
}
```

2.4.1. 1. d 2. d 3. a 4. d 5. d 6. b
7. c 8. a 9. b 10. b 11. c 12. c
13. c 14. b 15. c 16. a 17. a 18. d
19. a

2.4.2.

1. Este util şi în egală măsură elegant să declarăm un tip de date prin care să putem reprezenta coordonatele carteziene ale unui punct (înregistrare sau structură). Pentru a determina distanta minimă faţă de origine a unui punct, facem o parcurgere a fişierului; pentru a determina toate punctele aflate la această distanţă minimă, fie mai parcurgem odata fişierul, fie memorăm punctele într-un vector. Deoarece nu se cere distanţa minimă ci doar punctele aflate la respectiva distanţă, putem renunţa să extragem efectiv radicalul.

```
min←0; deschide fişier
citeşte n (din fişier)
┌pentru i←1,n execută
│   citeşte pi.x,pi.y (din fişier); d←pi.x*pi.x+pi.y*pi.y
│  ┌dacă d<min atunci min←d
│  └■
└■
închide fişier
┌pentru i←1,n execută
│  ┌dacă min= pi.x*pi.x+pi.y*pi.y atunci scrie(pi.x,pi.y)
│  └■
└■
```

2.

```
citeşte n
┌pentru i←1,n execută citeşte elev_i.nume,elev_i.mg
└■
┌pentru i←1,n-1 execută
│  ┌pentru j←i+1,n execută
│  │  ┌dacă elev_i.mg<elev_j.mg atunci
│  │  │  aux←elev_i;elev_i←elev_j;elev_j←aux
│  │  └■
│  └■
└■
┌pentru i←1,n execută scrie elev_i.nume,elev_i.mg
└■
```

3. Deorece numerele sunt ordonate după suma cifrelor se poate utiliza o metodă de rezolvare care să păstreze, alături de număr, cu ajutorul unei structuri, suma cifrelor acelui număr. Această incapsulare permite o prelucrare rapidă a datelor.

Varianta Pascal	*Varianta C/C++*
`Type elem=record`	`# include <iostream.h>`
`x,suma:integer end;`	`struct elem{ int x,suma;};`
`Var s:array[1..99] of elem;`	`elem s[100],aux;`
`aux:elem; y,n,i:integer;`	`int y,n,i,ordonat;`
`ordonat:boolean;`	`void main()`
`Begin Readln(n);`	`{ cin>>n;`
`for i:=1 to n do begin`	`for(i=1;i<=n;i++)`
`readln(s[i].x);`	`{cin>>s[i].x;`
`s[i].suma=0; y:=s[i].x;`	`y=s[i].x;`
`while y>0 do begin`	`s[i].suma=0;`
`s[i].suma:=s[i].suma+y mod 10;`	`while(y){s[i].suma+=y%10;`
`y:=y div 10 end;`	`y=y/10;}`
`y:=n;`	`y=n;`
`repeat ordonat:=true;`	`do{ ordonat=1;`

```
for i:=1 to y-1 do
 if(s[i].suma>s[i+1].suma) or
 (s[i].suma=s[i+1].suma) and
 (s[i].x>s[i+1].x) then
 begin aux:=s[i];
 s[i]:=s[i+1];s[i+1]:=aux;
 ordonat:=false
 end;
 y:=y-1
until ordonat;
for i:=1 to n do write(s[i].x,' ,)
end.
```

```
 for(i=1;i<y;i++)
 if(s[i].suma>s[i+1].suma) ||
  (s[i].suma==s[i+1].suma) &&
  (s[i].x>s[i+1].x)
   { aux=s[i];s[i]=s[i+1];
     s[i+1]=aux; ordonat=0;
    }
y--;
}while(!ordonat);
for(i=1;i<=n;i++)
 cout<<s[i].x<<' ';
}
```

3.

1. *Varianta Pascal*

```
var i,n,j:byte; f:text;
begin readln(n);
assign (f,'stelute.txt');rewrite(f);
for i:=1 to n do begin
   for j:=1 to i do write(f,'*');
   writeln(f)
  end;
for i:=n-1 downto 1 do begin
   for j:=i downto 1 do write(f,'*');
   writeln(f)
  end; close(f) end.
```

Varianta C/C++

```
#include <fstream.h>
void main()
{ int i,j,n;
  ofstream f("stelute.txt");
cin>>n;
for(i=1;i<=n;i++, f<<endl)
  for(j=1;j<=i; f<<'*',j++);;
for(i=n-1;i>=1;i--, f<<endl)
  for(j=1;j<=i; f<<'*',j++);;
f.close();
}
```

2. *Varianta Pascal*

```
var n,i,j:integer;
f:text;
begin readln(n);
  assign(f,'BAC.TXT'); rewrite(f);
  for i:=n downto 1 do begin
      for j:=i downto 1 do
         write (f,j,' ');
      writeln(f);
 end
close(f); end.
```

Varianta C/C++

```
#include <fstream.h>
void main()
{ int n,i,j;
  ofstream f(”BAC.TXT”);
  cin>>n;
  for(i=n;i>=1;i--)
     { for(j=i;j>=1;j--)
             f<<j<<" ";
       f<<endl;  }
  f.close();
}
```

3. *Varianta Pascal*

```
var n,i,j:integer; f:text;
begin
read(n);
assign(f,'BAC.TXT'); rewrite(f);
for i:=1 to n do begin
   for j:=i downto 1 do
      write(f,j,' ');
   writeln(f)
end;
close(f)
end.
```

Varianta C++/C

```
void main()
{int n,i,j;
ofstream f("BAC.TXT");
/FILE *f=fopen(”BAC.TXT”,”w”);
cin>>n;/scanf("%d",&n);
for(i=1;i<=n;i++)
  {for(j=i;j>=1;j--)
    f<<j<<" ";/fprintf(f,”%d ”,j);
    f<<endl; /fprintf(f,”\n”);
  }
f.close();/fclose(f); }
```

4. `citeste n` (de la tastatură)

```
pentru i←1,n execută scrie i*3 (în fişier, cu spaţiu)
■
```

5. Eficienţa metodei de rezolvare este evaluată prin utilizarea unui spaţiu minim de memorare. Sunt necesare două variabile **x** şi **y** care să memoreze capetele intervalului de intersecţie şi alte două variabile **a** şi **b** în care să citim capetele fiecărui interval. Valoarea variabilei **x** va fi cea mai mare dintre valorile primite de variabila **a** iar **y** va fi cea mai mică valoare dintre valorile primite de variabila **b**.

```
citeste n (din fişier)
x←0 ; y←999
pentru i←1,n execută
| citeste a,b (din fişier)
| dacă a>x atunci x←a
| ■
| dacă b<y atunci y←b
| ■
■
dacă x>y atunci
|  scrie MULTIME VIDA (pe ecran)
|  altfel scrie x,y (pe ecran)
■
```

6.

Varianta Pascal

```
var n:integer; f:text;
begin readln(n);
assign (f,'bac.txt');
rewrite(f);
if n mod 2<>0 then dec(n);
while n>=0 do   begin
    write(f,n,' ');dec(n,2) end;
close(f); end.
```

Varianta C/C++

```
#include <fstream.h>
void main()
{ int n,i;
  ofstream f("bac.txt");
  cin>>n;
  if (n%2)n--;
  for(i=n;i>=0;i-=2)f<<i<<' ';
 f.close(); }
```

7.

Varianta Pascal

```
var x,y:integer; f,g:text;
begin
assign(f,'numere.in');
assign(g,'numere.out');
reset(f); rewrite(g);
while not eof(f) do
begin readln(f,x,y);
 if x<y then
  writeln(g,x,' ',(x+y)/2:5:2,' ',y)
 else
  writeln(g,y,' ',(x+y)/2:5:2,' ',x)
 end;
 close(f); close(g)
end.
```

Varianta C

```
#include<stdio.h>
int main()
{int x,y; FILE *f, *g;
 f = fopen("numere.in","r");
 g = fopen("numere.out","w");
 while (!feof(f))
 {fscanf(f,"%d%d",&x,&y);
  if (x<y)
   fprintf(g,"%d %.1f %d\n",
       x,(x+y)/2.0,y);
  else
   fprintf(g,"%d %.1f %d\n",
       y,(x+y)/2.0,x);
 fcloseall(); }
```

8.

Varianta Pascal

```
var f,g:text;
    max_1,max_2,n,x,i:longint;
begin
 assign(f,'bac.in');reset(f);
 assign(g,'bac.out');rewrite(g);
 readln(f,n);
 max_1:=-maxlong; max_2:=-maxlong;
```

Varianta C++

```
#include <fstream.h>
#include <values.h>
long max_1,max_2,n,i,x;
void main()
{ifstream f("bac.in");
 ofstream g("bac.out");
 max_1=max_2=-MAXLONG;
```

```
for i := 1 to n do
 begin read(f,x);
  if x>max_1 then begin
      max_2:=max_1; max_1:=x end
  else if (x>max_2)and(x<max_1) then
          max_2:=x;
 end;
 writeln(g,max_1,' ',max_2);
 close(f); close(g);
end.
```

```
f>>n;
 for (i=0;i<n;i++)
 { f>>x;
   if(x>max_1)
   {max_2=max_1;max_1=x;}
   else
    if (x>max_2&&x<max_1)
       max_2=x; }
 g<<max_1<<' '<<max_2<<endl;
f.close(); g.close();}
```

9. Rezolvăm problema printr-o singură parcurgere a numerelor din fişier, fără să le reţinem într-un vector (la orice pas al algoritmului nu avem nevoie decât de numărul curent şi de cel precedent din fişier). Variabila **contor** va reţine lungimea secvenţei crescătoare curente. Comparând la fiecare pas valoarea variabile **contor** cu lungimea maximă determinată până în acel moment, rezolvăm şi situaţia când şirul de numere din fişier se termină cu o secvenţă strict crescătoare. Complexitate=**O(n)**, memorie folosită=câteva variabile simple.

Implementare pseudocod:

```
deschidere fişiere
 max←0;  contor←0;
 citeşte n (din fişier)
 citeşte a (din fişier)
 ┌pentru i←2,n execută
 │    citeşte x (din fişier)
 │    ┌dacă a<x atunci contor←contor+1
 │    │          altfel contor←0
 │    └■
 │    ┌dacă contor>max atunci max←contor
 │    └■
 │    a ← x
 └■
 scrie max
 închidere fişiere
```

10. Rezolvăm problema printr-o singură parcurgere a numerelor din fişier, fără să reţinem numerele într-un vector (la orice pas al algoritmului nu avem nevoie decât de numărul curent din fişier). Variabila contor va reţine lungimea secvenţei curente de valori strict pozitive; când citim o valoare negativă, atribuim variabilei contor valoarea `0`. Comparând la fiecare pas valoarea acestei variabile cu lungimea maximă determinată până în acel moment, rezolvăm şi situaţia când şirul de numere din fişier se termină cu o secvenţă de numere strict pozitive. Complexitate=**O(n)**, memorie folosită=câteva variabile simple.

Implementare pseudocod:

```
deschidere fişiere
max←0; contor←0
citeşte n (din fişier)
┌pentru i←1,n execută
│    citeşte x (din fişier)
│    ┌dacă x>0 atunci contor←contor+1
│    │          altfel contor←0
│    └■
│    ┌dacă contor>max atunci max←contor
│    └■
└■
scrie max
închidere fişiere
```

11. *Varianta Pascal*

```
var f,g:text;
    nr,n:integer;
begin
assign(f,'bac.txt');
reset(f);
assign(g,'cifre.txt');
rewrite(g);
while not eof(f) do
 begin
   read(f,nr);
   n:=abs(nr);
   if (n<=99)and(n>9)
      then
       write(g,nr,' ');
 end;
close(f);
close(g);
end.
```

Varianta C/C++

```
#include <fstream.h>
#include <math.h>
void main() {
fstream f("bac.txt", ios::in);
fstream g("cifre.txt", ios::out);
int nr,n;
 while (f>>nr)
 { n=abs(nr);
   if ((n<=99) &&(n>9))
        g<<nr<<' '; }
 f.close(); g.close();}
```

12. *Varianta Pascal*

```
var f,g:text;
    max_1,max_2,n,x,i:longint;
begin
 assign(f,'bac.in');
 reset(f);
 assign(g,'bac.out');
 rewrite(g);
 readln(f,n);
 max_1:=-maxlong;
 max_2:=-maxlong;
 for i := 1 to n do
 begin
  read(f,x);
  if x>max_1 then
   begin
      max_2:=max_1;
      max_1:=x
   end
  else if (x>max_2)and(x<max_1)
       then max_2:=x;
 end;
 writeln(g,max_1,' ',max_2);
 close(f);
 close(g);
end.
```

Varianta C++

```
#include <fstream.h>
#include <values.h>
long max_1,max_2,n,i,x;
void main()
{ifstream f("bac.in");
 ofstream g("bac.out");
 max_1=max_2=-MAXLONG;
 f>>n;
 for (i=0;i<n;i++)
 { f>>x;
   if(x>max_1)
   {max_2=max_1;max_1=x;}
   else
    if (x>max_2&&x<max_1)
       max_2=x;
 }
 g<<max_1<<' '<<max_2<<endl;
f.close();
g.close();
}
```

13. Citim pe rând numerele din fişier şi le afişăm pe ecran în formatul dorit. Pentru aceasta vom folosi un contor **k**, care ne va indica câte numere au fost scrise pe o linie. Simultan, numerele prime vor fi însumate

Varianta Pascal

```
assign(f,'bac.txt');
reset(f);  s:=0; k:=0;
 for i:=1 to 100 do begin
  readln(f,x); write(x,' ,');
  k:=k+1;
  if k=5 then
   begin writeln; k:=0; end;
  ok:=true; q:= trunc(sqrt(x));
  for j:=2 to q do
    if x mod i=0 then
     begin ok:=false;break end;
  if ok then s:=s+x;
 end;
 writeln;  write(s); close(f);
```

Varianta C/C++

```
ifstream f("bac.txt");
s=k=0;
for (i=1;i<=100;i++)
{f>>x; cout<<x<<' ';
 k++;
 if (k==5)
 { cout<<endl; k=0; }
 ok=1; q=sqrt(x);
 for (j=2;j<=q;j++)
   if (x%i==0)
      {ok=0;break;}
 if (ok) s+=x;
}
cout<<endl<<s; close(f);
```

14. Se determină poziţia (**pi**) primului număr par şi poziţia (**pf**) ultimului număr par în şirul de numere citit din fişier fără a memora numerele citite din fişier.

Varianta Pascal

```
var f:text;
......
assign (f,'BAC.TXT'); reset(f);
......
pi:=0; pf:=0; x:=0;
while not eof(f) do
 begin read(f,nr); x:=x+1;
  if nr mod 2=0 then
    begin
     if pi=0 then pi:=x;
     pf:=x
    end
  end;
writeln(pf-pi+1);
......
```

Varianta C/C++

```
fstream f("BAC.TXT", ios::in);
.......
pi=pf=0; x=0;
while (f>>nr)
  { x++;
    if (nr%2==0)
      {
        if (pi==0) pi=x;
        pf=x;
      }
  }
cout<<pf-pi+1;
......
```

15.

Varianta Pascal

```
var f,g:text;n,nr:integer;
......
assign (f,'DATE.IN'); reset(f);
assign (g,'DATE.OUT');
rewrite(g);
......
while not eof(f) do
 begin read(f,nr); n:=nr;
   while n>=10 do n:=n div 10;
   if n mod 2=0 then
        write(g,nr,' ');
 end; close(f); close(g); end.
```

Varianta C/C++

```
.........
fstream f("DATE.IN", ios::in);
fstream g("DATE.OUT", ios::OUT);
int n,nr;
 .......
 while (f>>nr)
   { n=nr;
     while(n>=10)n=n/10;
     if (n%2==0) g<<nr<<' ';
   }
 f.close(); g.close();
}
```

16. *Varianta Pascal*

```
var f:text; min,max,x,y:integer;
begin assign (f,'bac.txt'); reset(f);
read(f,max); min:=max;
while not eof(f) do
 begin read(f,x);
   if not eof(f)
   then begin read(f,y);
     if x>y then begin
       if x>max then max:=x;
       if y<min then min:=y; end
      else begin
       if y>max then max:=y;
       if x<min then min:=x; end; end
      else
     if x>max then max:=x
     else if x<min then min:=x;
 end;
writeln(max,' ',min); close(f); end.
```

Varianta C/C++

```
#include <fstream.h>
void main()
{fstream f("bac.txt", ios::in);
int max,min,x,y;
f>>max; min=max;
while (f>>x)
 if (f>>y)
   if (x>y)
     { if(x>max) max=x;
       if (y<min) min=y;}
   else
     { if(y>max) max=y;
       if (x<min) min=x;}
        else
        if (x>max) max=x;
         else if(x<min) min=x;
cout<<max<<" "<<min;
f.close(); }
```

17. Se determină mai întâi valoarea primului element al şirului şi de câte ori apare acesta. Apoi se vor afişa restul elementelor din şir.

```
p←[n/k]+1
┌pentru i←1,n%k executa scrie p
└■
┌pentru i←p-1,1,-1 executa
│   ┌pentru j←1,k executa scrie i
│   └■
└■
```

18. Citim pe rând câte un număr din fişier şi îl comparăm cu maximul parţial. Dacă valoarea nou citită este egală cu maximul parţial, vom incrementa contorul în care numărăm de câte ori apare maximul în fişier. Dacă valoarea citită este mai mare decât maximul parţial, atribuim valoarea sa acestui maxim, şi reiniţializăm contorul la 1. În pseudocod, algoritmul arată astfel:

```
deschide fisierul
citeste n; max←n; k←1
┌cat timp nu e sf. de fisier
│  citeste n
│   ┌daca n=max atunci k←k+1
│   │ altfel ┌daca n>max atunci  max←n; k←1
│   │        └■
│   └■
└■
scrie max, k
inchide fisier
```

19. Algoritmul în pseudocod de generare a primelor n numere prime este următorul:

```
scrie 2; k←1; i←3
┌cat timp k<n executa
│  ok←1
│  ┌pentru j←3,[i/2], pas 2 executa
│  │  ┌daca i%j=0 atunci
│  │  │ ok←0
│  │  └■
│  └■
│  ┌daca ok=1 atunci scrie i; k←k+1
│  └■
│  i←i+2
└■
```

20.

Varianta Pascal

```
var f,g:text; n,i,k:longint;
    nr:real;
function prim(n:longint):boolean;
begin
. . .
end;
begin assign(f,'bac.in');
reset(f);
assign(g,'bac.out');
rewrite(g);
read(f,n);
for i:=1 to n do begin
   read(f,nr);
 if prim(trunc(nr)) then
    begin
     write(g,nr:7:2,' ');
     inc(k);
    end;
 if k=0 then write(g,'NU EXISTA');
close(f);
close(g)
end.
```

Varianta C/C++

```
int prim(long n)
{ . . . }

void main()
{long n,i,k=0;
 float nr;
 ifstream f("bac.in");
 ofstream g("bac.out");
 f>>n;
 for(i=1;i<=n;i++)
  {f>>nr;
   if(prim(nr))
      {g<<nr<<" ";k++;}
   }
 if(k==0) g<<"NU EXISTA";
 f.close();
g.close();
}
```

21.

Varianta Pascal

```
var f,g:text;
    n,i,a,b,nr:integer;
begin
assign(f,'numere.in');
reset(f);
assign(g,'numere.out);
rewrite(f);
read(f,n,a,b);
for i:=1 to n do begin
 read(f,nr);
 if (nr<=a) or (nr>=b)
   then write(g,nr,' ');
close(f);close(g);
end.
```

Varianta C/C++

```
void main()
{int n,i,a,b,nr;
 ifstream f("numere.in");
 ofstream g("numere.out");
 f>>n>>a>>b;
 for(i=1;i<=n;i++)
  {f>>nr;
   if(nr<=a || nr>=b)
      g<<nr<<" ";
   }
 f.close();
 g.close();
}
```

22. *Varianta Pascal*

```
var f,g:text; n,i:integer;
    nr:longint;
begin assign(f,'bac.in'); reset(f);
assign(g,'bac.out');rewrite(f);
read(f,n);
for i:=1 to n do begin
 read(f,nr);
 if (nr>=1000)and(nr<=9999)
   then writeln(g,nr);
close(f); close(g);
end.
```

Varianta C/C++

```
void main()
{int n,i;long nr;
 ifstream f(„bac.in");
 ofstream g(„bac.out");
 f>>n;
 for(i=1;i<=n;i++)
  {f>>nr;
   if(nr>=1000&&nr<=9999)
      g<<nr<<endl;}
 f.close(); g.close(); }
```

23. *Varianta Pascal*

```
var f:text; n:longint; d:integer;
begin assign(f,'bac.out');
rewrite(f);
 for d:=100 to 999 do
  if n mod d = 0
    then write(f,d,' ');
 close(f)
end.
```

Varianta C/C++

```
void main()
{long n; int d;
 ofstream f("bac.out");
 cin>>n;
 for(d=100;d<1000;d++)
   if(n%d==0) f<<d<<" ";
 f.close();
}
```

24. *Varianta Pascal*

```
var f:text; i,n:integer;
  v:array[1..100] of integer;
begin
 assign(f,'vect.txt');
 rewrite(f);  read(n);
 for i:=1 to n do v[i]:= i mod 3;
 for i:=1 to n do write(f,v[i],' ');
 close(f);
end.
```

Varianta C/C++

```
#include<fstream.h>
void main()
{int v[100],i,n;
 ofstream f("vect.txt");
 cin>>n;
 for(i=0;i<n;i++) v[i]=i%3;
 for(i=0;i<n;i++) f<<v[i]<<" ";
 f.close();
}
```

25. Se parcurge fişierul o singură dată, calculându-se la fiecare pas suma secţiunii curente. Secţiunea curentă se termină în momentul în care valoarea citită din fişier determină o valoare pozitivă a sumei secţiunii; în acest caz, se începe o nouă secţiune, evident de suma 0. La fiecare pas, se compară suma secţiunii cu suma minimă determinată până în acel moment. Dacă fişierul conţine doar valori pozitive, atunci secţiunea de sumă minimă este formată dintr-un singur număr şi anume minimul din fişier!

```
citeşte n
 s←0; min←MAXINT; smin←0
┌pentru i←1,n execută
│  citeşte x din fişier
│ ┌dacă s+x>0 atunci s=0 altfel s=s+x
│ └■
│ ┌dacă s<min atunci min ←s
│ └■
│ ┌dacă x<min atunci min ←x
│ └■
└■
┌dacă min>0 atunci scrie min altfel scrie smin
└■
```

26. *Varianta Pascal*

```
var k,x:integer; f:text;
begin  assign(f,'numere.in');
 reset(f); k:=0;
 while not eof(f) do
  begin k:=k+1; read(f,x); end;
 close(f); writeln(k)
end.
```

Varianta C/C++

```
#include <fstream.h>
void main()
{  int k,x;
   ifstream f("numere.in");
   k=0;
   while(f>>x) k++;
   cout<<k; f.close();}
```

27.

```
deschide fişier f1
┌repetă
│   citeşte nr1;
│   deschide fişier f2
│  ┌repetă
│  │   citeşte nr2
│  │  ┌dacă  nr1=nr2  atunci contor←contor+1
│  │  └■
│  └■până când s-a detectat sfârşitul fişierului f2
│   închide fişer f2
│   scrie nr1,contor
└■până când s-a detectat sfârşitul fişierului f1
închide fişer f1
```

28.

```
deschide fişier
┌repetă
│   citeşte nr
│  ┌dacă  nr≥100 şi nr≤10000  atunci contor←contor+1
│  └■
└■până când s-a detectat sfârşitul fişierului
 scrie contor; închide fişier
```

29.

```
citeşte n
┌dacă n=1 atunci scrie 1 în fişier
│   altfel
│   k←1; p←1
│  ┌cât timp p≤n execută k←k+1;p←p*k
│  └■
│  ┌dacă p=n atunci scrie k-1 în fişier altfel
│  │   p2=p/(k-1)
│  │  ┌dacă p-n<n-p2 atunci scrie k-1 altfel scrie k-2
│  │  └■
│  └■
└■
```

Se poate avea în vedere şi următoarea soluţie: prin declararea unei constante cu tip (în Pascal) sau a unui vector cu valori iniţiale în C/C++ reţinem factorialele numerelor naturale de la 1 (sau 0 în C) până la 12 (13! are deja 10 cifre). Evident aceste factoriale se pot calcula „de mână" foarte repede (12 înmulţiri) şi vom atribui valorile în vector astfel încât $v_i=i!$. Tot ce avem de făcut în programul principal este să determinăm indicele elementului care minimizează diferenţa dintre el şi valoarea citită.

30. Se parcurg în ordine descrescătoare toate numerele naturale mai mici sau egale cu n şi se verifică care dintre acestea sunt divizori ai lui n.

```
deschide fişier
citeşte n
```

```
┌pentru j←n,1,-1 execută
│   ┌dacă n mod j=0 atunci scrie j în fişier
│   └■
└■
închide fişier
```

31. Se calculează suma numerelor din fişier iar apoi acestă sumă se scade din suma tuturor numerelor naturale de trei cifre. Rezultatul este egal cu numărul care lipseşte!

```
s←0; stoate←999*1000/2 - 99*100/2
deschide fişier f1(pentru citire)
┌pentru j←1,899 execută
│ citeşte nr din fişier; s←s+nr
└■
închide fişier f2
deschide fişier f2(pentru scriere)
scrie (stoate-s) în fişier
închide fişier f2
```

32. Întrucât toate numerele din fişier au cel mult două cifre fiecare, este suficient să utilizăm un **vector f** cu **100** de elemente (de la **0** la **99**), în care **f[i]=1** marchează prezenţa numărului **i** în fişier, în timp ce **f[i]=0** indică absenţa numărului **i** în fişier.

```
┌pentru i←1,2007*2007 execută
│  citeşte x (din fişier)
│  f[x]←1
└■
nr←0
┌pentru x←0,99 execută
│   ┌dacă f[x]=1 atunci
│   │   scrie x (pe ecran)
│   │   nr←nr+1
│   │   ┌dacă nr%5=0 atunci
│   │   │ salt la linie nouă
│   │   └■
│   └■
└■
```

33. Se observă că prima fracţie din şir este $\frac{0}{1}$, iar ultima $\frac{1}{1}$. Aceste perechi se scriu separat în fişier. Toate celelalte perechi se generează conform secvenţei pseudocod:

```
citeste n;
a←0;
b←1;
x←1;
y←n
┌cat timp x>1 sau y>1 executa
│z←[(a+n)/y]*x-a; u←[(a+n)/y]*y-b; scrie z,u
│a←x;b←y;x←z;y←u
└■
```

34. Poziţia unei litere mici, **l**, în alfabetul englez poate şi calculată cu ajutorul codului ASCII: **ord(l)-ord('a')+1 | l-'a'+1.**

```
citeste n (din fisier)
┌pentru i<1,n executã
│   citeste l
│   scrie codAscii(l)-codAscii('a')+1
└■
```

35. *Varianta Pascal*

```
var f:text; e:integer; c,x:char;
Begin
assign (f,'BAC.TXT');
reset(f); read(f,x);
e:=ord(x)-ord('0');
write(E);
while not(eof(f)) do begin
read(f,c);
if c<>'=' then begin
 read(f,x); write(c,x);
 if c='+'
   then E:=E+ ord(x)-ord('0')
   else E:=E-ord(x)+ord('0');
 end
 end;
  write(' = ',E); close(f);
end.
```

Varianata C/C++

```
#include <fstream.h>
void main()
{
fstream f("BAC.TXT", ios::in);
int e,x; char c;
f>>e;cout<<e;
while(f>>c)
{
  if(c!='=')
   { f>>x; cout<<c<<x;
     if(c=='+')e=e+x;
          else e=e-x;
   }
 }
 cout<<" = "<<e;
 f.close();
}
```

36. *Varianta Pascal*

```
var f,g:text; k,i:longint;
    v:array[0..9] of byte;
Begin
 assign (f,'DATE.IN'); reset(f);
 assign (g,'DATE.OUT');
 rewrite(g);
 while not eof(f) do
  begin read(f,i);
   repeat
    inc(v[i mod 10]); i:=i div 10;
   until i=0;
  end;
 for i:=0 to 9 do
    if v[i]=0 then inc(k);
 write(g,k); close(f); close(g)
end.
```

Varianta C/C++

```
#include <fstream.h>
#include <math.h>
void main()
{ fstream f("DATE.IN", ios::in);
  fstream g("DATE.OUT", ios::out);
  int v[10];long i,k=0;
  for(i=0;i<=9;i++)v[i]=0;
  while (f>>i)
     do{v[i%10]++; i=i/10;
       }while(i);
  for(i=0;i<=9;i++)
     if (v[i]==0)k++;
  g<<k;
  f.close(); g.close();
}
```

37. Putem rezolva problema printr-o singură parcurgere a numerelor din fişier; trebuie să fim atenţi la două cazuri particulare şi anume, vârful crestei să nu fie pe prima sau pe ultima poziţie. Pentru aceasta, citim primele două elemente din fişier şi dacă primul e mai mare decât al doilea, şirul nu are proprietatea cerută. Dacă nu, citim câte un număr din fişier în variabila **x** şi îl comparăm cu cel anterior memorat în variabila **b**; distingem următoarele cazuri (în care variabila **s** are valoarea **1** dacă suntem pe partea crescătoare a şirului şi **-1** dacă suntem pe partea descrescătoare, adică dacă am găsit deja o pereche de valori vecine care îndeplinesc condiţia că prima este mai mare decât a doua):

- dacă **b<x** şi **x** a fost ultimul număr din fişier, nu e ok;
- dacă **b<x** şi **x** nu a fost ultimul număr din fişier, atunci dacă **s=-1** înseamnă că suntem pe panta descrescătoare şi nu e bine; dacă **s=1**, citim următorul număr;
- dacă **b>x** atunci, dacă a fost prima oară când e îndeplinită această condiţie, e posibil ca **b** să fie maximul din fişier, deci îi reţinem valoarea, schimbăm valoarea variabilei **s** (**s←-1)** şi citim următorul număr.

În continuare sunt date posibile implementări ale acestui algoritm:

Varianta Pascal

```
var fi,fo:text;
    max,n,x,a,b,i,s:integer;
procedure err;
begin writeln(fo,0);
      close(fo); close(fi); end;
begin assign(fi,'bac.in');reset(fi);
 assign(fo,'bac.out');rewrite(fo);
 readln(fi,n); read(fi,a,b);
 if a>b then begin err;exit end;
 s:=1;
 for i := 3 to n do
  begin read(fi,x);
  if b<x then begin
   if i=n then  begin err;exit end;
   if s=-1 then begin err;exit end;
  end
  else if s=1 then
          begin max:=b; s:=-1 end;
  b := x
 end;
  writeln(fo,max);close(fo); close(fi)
end.
```

Varianta C++

```
#inlcude <fstream.h>
int max,n,i,x,a,b,s=1;
void main()
{ifstream fi("bac.in");
 ofstream fo("bac.out");
 fi>>n;
 fi>>a>>b;
 if (a>b) {fo<<0;return;}
 for (i=3;i<=n;i++)
 {fi>>x;
  if (b<x)
  {if (i==n){fo<<0;return;}
   if(s==-1){fo<<0;return;}
  }
   else  /* b>x */
   if (s==1) /* e prima data*/
   { max=x; s=-1;}
   b=x;
 }
 fo<<max<<endl;
}
```

38. Rezolvăm problema printr-o singură parcurgere a numerelor din fişier: determinăm simultan de câte ori apare fiecare dintre valorile care teoretic pot să apară în fişier (valorile de la **1** la **500**). Pentru aceasta ne trebuie **500** contori, deci un vector pe care îl denumim **ct** cu **501** elemente (şi nu **500** pentru a nu complica inutil lucrurile în C/C++); în acesta, **ct[i]** va reprezenta de câte ori apare valoarea **i** în fişier (pentru exemplul din enunţ, **ct[6]** va avea valoarea **3**, **ct[7]** va avea valoarea **2**, **ct[8]** va avea valoarea **0** etc.). Rezultatul îl obţinem parcurgând acest vector crescător şi afişând poziţia şi valoarea elementele diferite de **0**. Algoritmul este liniar iar spaţiul de memorie utilizat este evident mai mic decât într-un algoritm în care am memorat într-un vector toate valorile din fişier pe care să îl folosim pentru a rezolva problema.

Implementare pseudocod:

```
fi : fişier de intrare
fo : fişier de ieşire
ct : vector de 501 elemente întregi cu indicele de la 0
i  : variabilă întreagă

deschidere fişiere
┌pentru i←0,500 execută ct[i] ←0;
│┌cât timp nu se atinge sfârşitul fişierului fi execută
│||    citeşte x (din fişier)
│||    ct[i] ← ct[i]+1
│└■
└■
┌pentru i←0,500 execută
│  ┌dacă ct[i] ≠ 0 atunci
│  │  scrie (i,ct[i]) în fişierul fo (în formatul din enunţ!!)
│  └■
└■
închidere fişiere
```

39. Citim primul şir din fişier în vectorul a; prin căutare binară, determinăm apoi valorile comune primelor două şiruri – deci intersecţia lor; memorăm această interseţie tot în vectorul a, şi folosim aceeaşi metodă pentru a face intersecţia între noul conţinut a lui a şi al treile-a şir şamd. Putem să rezolvăm problema şi cu un algoritm asemănător celui de interclasare. Deoarece ambele metode folosesc faptul că elementele şirurilor sunt sortate, ambele sunt bune din punct de vedere al eficienţei, chiar dacă au complexităţi diferite.

Varianta Pascal

```
type sir=array[1..2000] of integer;
var f:text;a:sir; n:integer;
procedure citire ( var a:sir;
                   var n:integer);
var x:integer;
begin  n:=0;  read(f,x);
  while x<>-1 do begin
    inc(n);a[n]:=x;read(f,x); end
end;
procedure afisare(a:sir;n:integer);
var i:integer;
begin
 for i:=1 to n do write(a[i],' ');
end;
function cautbin(a:sir;
     n,x:integer):boolean;
var i,j,m:integer;
begin i:=1;j:=n; cautbin:=false;
  while i<=j do begin
     m:=(i+j) div 2;
     if a[m]=x then
      begin cautbin:=true;break end
     else if a[m]>x then j:=m-1
          else i:=m+1
  end
end;
procedure intersectie
         (var a:sir; var n:integer);
var c:sir; l,x,i,k:integer;
begin read(f,x); l:=0;
  while x<>-1 do begin
     if cautbin(a,n,x) then
       begin inc(l);c[l]:=x; end;
      read(f,x); end;
  n:=l;  a:=c;
end;
begin  assign(f,'bac.txt');
 reset(f); citire(a,n);
 while not eof(f) do begin
   intersectie(a,n);
   if n=0 then begin
      write('inters. vida ');
      break
     end
 end; afisare(a,n);
end.
```

Varianta C/C++

```
ifstream f("BAC.TXT");
void citire(int a[],int &n)
{ int x; n=0; f>>x;
  while(x!=-1)
     {a[n]=x;    n++; f>>x; }
}
void afisare(int a[],int n)
{int i;
 for(i=0;i<n;i++)cout<<a[i]<<" ";
}

int cautbin(int a[],int n,int x)
{int i=0,j=n-1,m;
 while(i<j)
   { m=(i+j)/2;
     if(x==a[m])return 1;
     else if(x<a[m]) j=m-1;
          else i=m+1; }
 return 0;
}
void intersectie(int a[], int &n)
{ int c[2000],l=0,x,k,i;
  f>>x;
  while(x!=-1)
  { if(cautbin(a,n,x)
       {c[l]=x; l++;}
    f>>x; }
  n=l;
  for(i=0;i<n;i++) a[i]=c[i];
}
void main()
{int a[2000],n;
 citire(a,n);
 while(!f.eof())
  {intersectie(a,n);
   if(n==0)
   {cout<<"inters. vida";
    break;}
  }
afisare(a,n);
}
```

40. Vom utiliza un vector caracteristic **c** cu **100** de elemente, cu indici de la **0** la **99**. Fiecare element acestui vector, **c[x] (0≤x≤99)** va indica numărul de apariţii ale valorii **x** în fişier. Următorul pas este determinarea valorii maxime din vectorul **c**, aceasta semnificând numărul maxim de apariţii a unei valori în fişier. În final se parcurge din nou vectorul **c** şi se afişează indicii acelor elemente care sunt egale cu valoarea maximă determinată anterior.

Varianta Pascal

```
var c:array[0..99]
     of integer;
    n,x,i,max:integer;  f:text;
begin
assign(f,'NUMERE.TXT');
reset(f);
readln(f,n);
for i:=1 to n do begin
    read(f,x); inc(c[x])
end;
max:=c[0];
for x:=1 to 99 do
  if c[x]>max then max:=c[x];
for x:=0 to 99 do
  if c[x]=max then
               write(x,' ');
close(f)
end.
```

Varianta C/C++

```
void main()
{int c[100],n,x,i,max;
 FILE
*f=fopen("NUMERE.TXT","r");
 /ifstream("NUMERE.TXT");
 for(x=0;x<=99;x++)  c[x]=0;
 fscanf(f,"%d",&n); / f>>n;
 for(i=1;i<=n;i++)
  {fscanf(f,"%d",&x); / f>>x;
   c[x]++;  }
 max=c[0];
 for(x=1;x<=99;x++)
   if(c[x]>max)   max=c[x];
 for(x=0;x<=99;x++)
   if(c[x]==max)
     printf("%d ",x);
    /cout<<x<<" ";
}
```

41. Pentru a afişa numărul cerut utilizând un număr minim de operaţii se completează un vector cu **10** elemente, fiecare element **f[i]** reprezentând frecvenţa apariţiei cifrei **i** în fişier.După completare, se parcurge acest vector descrescător („de la dreapta la stânga") şi se afişează fiecare cifră **i** de **f[i]** ori.

Varianta Pascal

```
var f:array['0'..'9']of integer;
    fis:text;
    cifra:char;
    i:integer;
Begin assign(fis,'bac.txt');
for i:=1 to 2000 do begin
 read(fis,c);
 f[c]:=f[c]+1 end;
close(f);
for c:='9' downto '0' do
  while(f[c]>0) do begin
    write(c);f[c]:=f[c]-1
  end
end.
```

Varianta C/C++

```
# include <fstream.h>
ifstream fis("bac.txt");
int f[10],i;
void main()
{char cifra;
 for(i=1;i<=2000;i++)
 {fis>>cifra;f[cifra-'0']++; }
  fis.close();
  for(i=9;i>=0;i--)
   while(f[i]){cout<<i; f[i]--;}
}
```

42. Principalii paşi ai algoritmului de rezolvare a acestei probleme sunt următorii:

- **asignarea/deschiderea fişierului pentru citire**
- **citirea valorii lui n de pe prima linie a fişierului**

m←0

```
┌pentru i←1,n execută
│ citeste x; ok←true
│   ┌pentru j←2,sqrt(x) executa
│   │   ┌daca x%j=0 atunci ok←false
│   │   └■
│   └■
│   ┌daca ok=true atunci m←m+1; a[m]←x
│   └■
└■
închiderea fişierului
sortarea vectorului a[i], i=1,m folosind orice metodă cunoscută
┌pentru i←1,m execută scrie a[i]
└■
```

Se observă că se vor memora doar numerele prime într-un vector, sortarea fiind astfel mai rapidă, fiind vorba de mai puţine elemente de sortat. Transcrierea acestui algoritm în limbaj de programare, rămâne ca exerciţiu.

43. În pseudocod algoritmul arată astfel:

```
citeste n; m←0
┌pentru i←1,n executa citeste x[i]; m←m+x[i]
└■
m←[m/n]
┌pentru i←1,n executa
│   ┌daca x[i]=m atunci  k←k+1
│   └■
└■
scrie k
```

44. Se va folosi un vector cu elemente de la **0** la **9999** în care **v[i]=0** dacă valoarea i nu se găseşte în fişier, respectiv **v[i]=1** dacă s-a găsit în fişier. Eficienţa constă în faptul că nu avem nevoie de algoritm de căutare sau de sortare. Algoritmul în pseudocod arată astfel:

```
deschide fisierul
citeste n
┌pentru i←0,9999 executa v[i]←0
└■
┌pentru i←1,n executa citeste x; v[x]←1
└■
┌pentru i←0,9999 executa
│   ┌daca v[i]=1 atunci scrie i
│   └■
└■
inchide fisierul
```

45. *Varianta Pascal*

```
var f:text; m,n,k,i,j,x:integer;
    ok:boolean;
    a:array[1..50] of integer;
begin  assign(f,'bac.txt');
 reset(f); k:=0; readln(f,m,n);
 for i:=1 to m do
 begin ok:=true;
  for j:=1 to n div 2 do
    read(f,a[j]);
  if n mod 2=1 then read(f,x);
  for j:=n div 2 downto 1 do
   begin read(f,x);
     if x<>a[j] then ok:=false;
   end;
  if ok then k:=k+1;
  readln(f);
 end;
 close(f); writeln(k)
end.
```

Varianta C/C++

```
#include <fstream.h>

main()
{  ifstream f("bac.txt");
   int ok,m,n,k,i,j,x,a[50];
   k=0; f>>m>>n;
   for(i=0;i<m;i++)
   { ok=1;
     for (j=0;j<n/2;j++)
      f>>a[j];
     if (n%2==1) f>>x;
     for (j=n/2-1;j>=0;j--)
     { f>>x;
       if (x!=a[j]) ok=0;
     }
     if (ok) k++;
   }
   f.close(); cout<<k;
}
```

46. a) Este suficient să păstrăm în memorie doar cele `k` numere care formează secvenţa curentă. Completarea vectorului care reţine aceste numere se face circular, începând cu poziţia `0`. Orice nou element situat pe poziţia **`i(>=k)`** în şirul celor `n` numere va fi păstrat pe poziţia **`i%k`** timp de `k` paşi (el intră în componenţa a **`k`** secvenţe consecutive). Această metodă este eficientă din punct de vedere al timpului de executare (complexitate timp ***O(n)***) întrucât se face o singură parcurgere a şirului de valori (trecere absolut necesară) iar din punct de vedere al spaţiului alocat este eficientă pentru că se reţin în vector doar **`k`** valori (complexitate spaţiu ***O(k)***).). O altă posibilitate de rezolvare, la fel de eficientă, dar care presupune un efort mai mare de programare se bazează pe implementarea unei cozi alocată dinamic în locul vectorului completat circular.

b) *Varianta Pascal*

```
var SIR:array[0..98] of integer;
    n,i,k:word; max,s:longint;
    f:text;
Begin readln(k);
assign(f,'bac.txt');reset(f);
readln(f,n);
for i:=0 to k-1 do begin
  read(f,SIR[i]);s:=s+SIR[i]
end;
max:=s;
for i:=k to n-1 do begin
   s:=s-SIR[i mod k];
   read(f,SIR[i mod k]);
   s:=s+SIR[i mod k];
   if s>max then max:=s
end;
write(max);
close(f)
end.
```

Varianta C/C++

```
# include <fstream.h>

int SIR[99]; unsigned n,i,k;
long max,s;
ifstream f("bac.txt");
void main()
{ cin>>k; f>>n;
  for(i=0;i<=k-1;i++)
   {f>>SIR[i];s=s+SIR[i];}
   max=s;

for(i=k;i<=n-1;i++)
 { s=s-SIR[i%k]; f>>SIR[i%k];
   s=s+SIR[i%k];
   if(s>max)max=s;
  }
cout<<max);
f.close();
}
```

47. Este suficient să utilizăm un tablou unidimensional **f** cu **10** elemente, fiecare componentă **f[i] (i∈[0;9])** reprezentând numărul aparițiilor cifrei **i** în şirul numerelor din fişier. Determinarea valorii maxime din vectorul **f** şi afişarea indicilor **i** pe care se găseşte această valoare încheie rezolvarea problemei.

```
pentru i=0,9 execută f[i]←0
■
cât timp nu am ajuns la sfârşitul fişierului execută
 citeste x (din fişier)
 cât timp x>0 executa
  f[x%10]←f[x%10]+1   (x%10 reprezintă restul împărţirii numărului x la 10)
  x←[x/10]
 ■
■
m←0
pentru i=0,9 execută
  daca f[i]>m atunci m←f[i]
  ■
■
pentru i=0,9 execută
  daca f[i]=m atunci scrie i (pe ecran)
  ■
■
```

Se poate evita secvenţa de prelucrare a cifrelor unui număr dacă citim din fişier într-o variabilă de tip **char**. În acest caz, trebuie să fim atenţi cum declarăm vectorul **f**.

48. *Varianta Pascal*

```
var f:text; x,nr:longint; k:byte;
function prim(n:longint):boolean;
 var d:longint;
 begin prim:=true;
  if n<2 then prim:=false
  else
  for d:=2 to trunc(sqrt(n)) do
      if n mod d=0 then    begin
          prim:=false; break end;
 end;
function cifra
      (n:longint; k:byte):boolean;
 begin cifra:=false;
  if n=k then cifra:=true
  else
   repeat
    if n mod 10=k then
     begin cifra:=true; break; end
     else n:=n div 10;
   until n=0;
 end;
begin
 readln(k);
 assign (f,'DATE.IN');
 reset(f);
```

Varianta C/C++

```
#include <fstream.h>
#include <math.h>

int prim(long n)
{ if (n<2) return 0;
  long d;
  for(d=2;d*d<=n;d++)
     if (n%d==0) return 0;
 return 1;}

int cifra(long n, int k)
{ if (n==k) return 1;
  do { if (n%10==k) return 1;
       n=n/10;
     } while (n>0);
 return 0;
}
void main()
{ fstream f("DATE.IN", ios::in);
  int k; long x,nr=0; cin>>k;
  while (f>>x)
   if (prim(x) && cifra(x,k))nr++;
  cout<<nr; f.close();
}
```

```
while not eof(f) do
    begin
     read(f,x);
     if prim(x) and cifra(x,k)
        then inc(nr);
    end;
 write(nr);
 close(f);
end.
```

49. Vom folosi un vector de frecvenţă, în care numărăm de câte ori apare fiecare cifră în scrierea numerelor din fişier. Citirea din fişier o vom face caracter cu caracter.

Varianta Pascal

```
program bac3;
var c:char; n,k:integer;
 f:text;
 a:array['0'..'9'] of integer;
begin
 assign(f,'numere.in');
 reset(f); readln(f,n);
 while not eof(f) do
  begin
   read(f,c);
   if c in ['0'..'9'] then
    begin
     if a[c]=0 then k:=k+1;
     a[c]:=a[c]+1;
    end;
 close(f);
 if k=10 then
     writeln('NICI UNA')
 else
  for c:='0' to '9' do
    if a[c]=0 then
      write(c,' ');
end.
```

Varianta C/C++

```
#include <fstream.h>

void main()
{  int k,n; int a[10];
   char c;
   ifstream f("numere.in");
   k=0;
   f>>n;
   for (c='0';c<='9';c++)
     a[c-'0']=0;
   while(f>>c)
   {  if ((c>='0')||(c<='9'))
      { if (a[c-'0']==0) k++;
        a[c-'0']++;
      }
   }
   f.close();
   if (k==10) cout<<"NICI UNA";
   else
     for (c='0';c<='9';c++)
       if (a[c]==0) cout<<c<<' ';
}
```

50. O soluţie pentru calculul sumei celor mai mari **m** elemente negative din fişier ar consta în memorarea într-un vector a elementelor negative din fişier, ordonarea descrescătoare a vectorului iar apoi însumarea primelor **m** elemente şi re-deschiderea fişierului astfel încât să putem adăuga informaţii în el. Evident, trebuie să se verifice dacă numărul elementelor negative din fişier este suficient pentru calculul sumei, dacă este mai mare decât **m**.

```
deschide fişier
citeşte n,m (din fişier)
k←0
┌pentru i←1,n execută
│   citeşte nr (din fişier)
│   ┌dacă nr<0 atunci k←k+1; v_k←nr
│   └■
└■
închide fişier
```

```
┌dacă k<m atunci afişează 0
│   altfel
│     ordonare(v,k);  S←0;
│     ┌pentru i←1,m execută S←S+v_i
│     └■
│    afişează S
└■
```

51. O variantă posibilă de rezolvare constă în utilizarea unui vector cu p componente, în fiecare componentă **i** se va contoriza numărul de elevi care poartă uniformă de tip **i**.

```
deschide fişier
citeşte n,p (din fişier)
┌pentru i←1,n execută
│   citeşte nr,i (din fişier)
│   Tip_unif_i← Tip_unif_i+nr
└■
┌pentru i←1,p-1 execută
│  ┌pentru j←i+1,p execută
│  │  ┌ dacă  Tip_unif_i < Tip_unif_j atunci
│  │  │   aux ←Tip_unif_i;
│  │  │   Tip_unif_i←Tip_unif_j;
│  │  │   Tip_unif_j←aux;
│  │  └■
│  └■
└■
┌pentru i←1,n execută
│   scrie Tip_unif_i
└■
```

52. O variantă posibilă de rezolvare constă în determinarea numărului de grupe complete (în care apar toate cifrele de la **1** la **9)**, iar apoi determinarea cifrelor din ultima grupa incompletă.

```
deschide fişier(pentru scriere)
citeşte n
grupe←n div 10;
rest←n mod 10
```

```
┌pentru i←1,grupe execută
│  ┌pentru j←1,9 execută scrie j
│  └■
└■
┌pentru j←1,rest execută scrie j
└■
```

```
53.    citeşte n,m
       deschide fişierul
       ┌pentru i←1,n execută
       │  ┌pentru j←1,m execută citeşte a_ij
       │  └■
       └■
       închide fişier
       ┌pentru i←1,n execută
       │  ┌pentru j←1,m execută afişează a_ij
       │  └■
       │   salt la linie nouă
       └■
```

54. Considerând indicii liniilor şi coloanelor variind în intervalul **[1,n]**, observăm că fiecare element al matricei

$$Aij=\begin{cases}(i+j-1)\%n, & \text{dacă } (i+j-1)\%n\neq 0\\ n, & \text{dacă } (i+j-1)\%n=0\end{cases}$$

55. *Varianta Pascal*

```
var f,g:text;
    k:longint; i:byte;
    v:array[0..99] of byte;
Begin
 assign (f,'DATE.IN'); reset(f);
 assign (g,'DATE.OUT');
 rewrite(g);
 while not eof(f) do
   begin
    read(f,i); inc(v[i]);
   end;
 for i:=0 to 99 do
    if v[i]=1 then inc(k);
 write(g,k);
 close(f); close(g)
end.
```

Varianta C/C++

```
#include <fstream.h>
#include <math.h>
void main()
{ fstream f("DATE.IN", ios::in);
  fstream g("DATE.OUT", ios::out);
  int v[100],i;long k=0;
  for(i=0;i<=99;i++)v[i]=0;
  while (f>>i)v[i]++;
  for(i=0;i<=99;i++)
     if (v[i]==1)k++;
  g<<k;
  f.close(); g.close();}
```

56. Una dintre soluţiile posibile este cea în care se completează direct elementele aflate sub diagonala principală.

Putem observa că:

$$Aij=\begin{cases} i-j+1, & i>=j \\ 0, & i<j \end{cases}$$

```
citeste n ( de la tastatură)
┌pentru i←1,n execută
│ ┌pentru j←1,n execută
│ │ ┌dacă i>=j atunci Aij←i-j+1
│ │ │ altfel Aij←0
│ │ └■
│ └■
└■
┌pentru i←1,n execută
│ ┌pentru j←1,n execută
│ │ scrie Aij
│ │    (în fişier, cu spaţiu)
│ └■
│ salt la linie nouă (în fişier)
└■
```

57. Se memorează într-un vector doar elementele de pe conturul matricei, această memorare facându-se pe parcursul citirii datelor din fişier. Se ordonează apoi crescător vectorul astfel construit şi se afişează apoi elemntele sale pe ecran.

```
deschide fişierul(pentru citire)
citeşte m,n din fişier
dim←0
┌pentru i←1,m execută
│  ┌pentru j←1,n execută
│  │   citeşte nr
│  │  ┌dacă i=1 sau i=m sau j=1 sau j=n atunci dim←dim+1; a_dim←nr
│  │  └■
│  └■
└■
închide fişier;
ordonare(a,dim);
afişare(a,dim)
```

58. Se ştie că elementul **a[i,j]** este pe diagonala principală dacă **i=j**, este pe diagonala secundară dacă **i+j=n+1**. Elementele din zona **A** sunt cele pentru care **j>i** şi **j<n-j+1**. Elementele din zona **B** sunt cele pentru care **j>n-i+1** şi **j<i** Algoritmul pseudocod care este următorul:

```
citeste n
┌pentru i←1,n execută
│ ┌pentru j←1,n executa
│ │ a[i,j]=3
│ │ ┌daca i=j sau i+j=n+1 atunci a[i,j]=1
│ │ │  altfel
│ │ │ ┌daca (j>i si j<n-j+1) sau (j<i si j>n-i+1) atunci a[i,j]=2
│ │ │ └■
│ │ └■
│ └■
└■
deschide fisierul
┌pentru i←1,n executa
│ ┌pentru j←1,n executa scrie a[i,j] in fisier
│ └■
│ scrie rand nou in fisier
└■
inchide fisierul
```

59. Intrucât trebuie să selectăm doar elementele comune celor două şiruri este necesară memorarea acestora. Se pot memora ambele şiruri sau putem reţine doar numerele din primul şir şi să căutăm secvenţial fiecare valoare din cel de-al doilea şir.

Dacă presupunem că am memorat şi ordonat crescător elementele celor două şiruri, primul de dimensiune n şi cel de-al doilea de dimensiune m, atunci o parcurgere simultană este suficientă pentru determinarea valorilor cerute.

```
i←1; j←1
┌cat timp i<=n si j<=m executa
│ ┌daca a_i=b_j atunci scrie a_i;  i←i+1;  j←j+1
│ │ altfel ┌daca  a_i<b_j atunci i←i+1
│ │        │            altfel j←j+1
│ │        └■
│ └■
└■
```

60. Se pleacă de la elementul de pe linia **L** coloana **C**, în sus până la întâlnirea marginii matricei, adică până când linia este **<1** sau coloana este **>N**. Deplasarea în sus se face scăzând o unitate la linie şi crescând coloana cu o unitate. Similar vom proceda pentru cealaltă parte a semidiagonalei, parcurgînd-o în jos până la marginea matricea. Algoritmul de rezolvare a problemei este următorul.

```
i←L; j←C; s←0
┌cat timp i>=1 si j<=N executa s←s+a[i,j]; i←i-1; j←j+1
└■
i←L+1; j←C-1
┌cat timp i<=N si j>=1 executa s←s+a[i,j]; i←i+1; j←j-1
└■
scrie s
```

61. Vă propunem una dintre soluţiile posibile: parcurgem elementele din matrice aflate pe cele două linii **i** şi **k** ale ramei şi le însumăm în variabila **S**. Parcurgem apoi elementele matricei aflate pe cele două coloane **j** şi **l**, având grija ca de această dată să nu luăm în considerare primul şi ultimul element al fiecărei linii deoarece ele au fost deja incluse în suma calculată la pasul anterior.

```
deschide fişier
citeşte n,m (din fişier)
┌pentru i←1,n execută
│  ┌pentru j←1,m execută citeşte a(i,j) (din fişier)
│  └■
└■
închide fişier; citeşte i,j,k,l; S←0
┌pentru indice←j,l execută
│   S←S+ a(i,indice)
│  ┌dacă i≠k atunci S←S+ a(k,indice)
│  └■
└■
┌pentru indice←i+1,k-1 execută
│   S←S+ a(j,indice)
│  ┌dacă j≠l atunci S←S+ a(i,ndicel)
│  └■
└■
afişează S
```

Cu un mic efort, s-ar putea renunţa la memorarea valorilor într-o matrice, calculând suma cerută pe măsură ce se parcurge fişierul.

62.

Varianta Pascal	*Varianta C/C++*

`var f:text;`
`......`
`assign (f,'BAC.TXT'); rewrite(f);`
`......`
`m:=length(s);`
`for i:=m downto 1 do`
`begin for j:=1 to i do write(f,s[j]);`
`writeln(f)`
`end;`
	`fstream f("BAC.TXT", ios::out);`
	`.......`
	`m=strlen(s);`
	`for(i=m-1;i>=0;i--)`
	`{`
	`for(j=0;j<i;j++)`
	`f<<s[j];`
	`f<<endl;`
	`}`

63. Variabila **s** va reţine şirul de caractere citit de la tastatură.

Varianta Pascal	*Varianta C/C++*

`var f:text;`
`......`
`assign (f,'BAC.TXT'); rewrite(f);`
`......`
`k:=length(s);`
`for i:=1 to k do begin`
`for j:=k-i+1 to k do`
`write(f,s[j]);`
`writeln(f); end;`
`.......`
	`fstream f("BAC.TXT",ios::out);`
	`.......`
	`k=strlen(s);`
	`for(i=1;i<=k;i++)`
	`{`
	`for(j=k-i+1;j<=k;j++)`
	`f<<s[j-1];`
	`f<<endl;`
	`}`
	`.......`

64. ***Varianta Pascal***

```
var f:text; s:string;
    c:char;n,i:integer;
Begin assign (f,'BAC.TXT');
rewrite(f);
readln(s); readln(c);
n:=length(s);
for i:=1 to n do
  begin
   write(f,s[i]);
   if c=s[i] then write(f,c);
  end;
close(f) end.
```

Varianta C/C++

```
#include <fstream.h>
#include <string.h>
void main()
{ fstream f("BAC.TXT",ios::out);
  char s[31],c;
  int i,n;
  cin.get(s,31);cin>>c;
  n=strlen(s);
  for(i=0;i<n;i++)
    { f<<s[i];
      if(s[i]==c)f<<c; }
  f.close();}
```

65. ***Varianta Pascal***

```
var f:text; s:string;
    c:char;x,i:byte;
Begin
assign (f,'DATE.IN');reset(f);
readln(s); x:=length(s);
while (not eof(f))and (x>0) do
  begin
    read(f,c); i:=pos(c,s);
    while i*x>0 do
     begin delete(s,i,1);
        i:=pos(c,s); dec(x); end;
  end;
  if x=0 then write('DA')
         else write('NU');
close(f)
end.
```

Varianta C/C++

```
#include <fstream.h>
#include <string.h>
void main()
{ fstream f("DATE.IN",ios::in);
  char s[51],c,*p;  int x;
  cin.get(s,51); x=strlen(s);
  while((f>>c)&&x)
  { cout<<c<<" "<<s<<endl;;
    p=strchr(s,c);
    while(p)
     { strcpy(p,p+1);
       p=strchr(p,c);x--;}
  }
  if(x==0)cout<<"DA";
     else cout<<"NU";
 f.close();}
```

66. Algoritmul constă în citirea pe rând a câte unui prenume din fişier şi de a-l compara cu lungimea maximă determinată până în momentul respectiv. În caz de egalitate cu acesta vom mări contorul pentru numărul de prenume de lungime maximă, iar dacă noua valoare este mai mare decât maximul parţial, vom actualiza valoarea maximului şi iniţializăm contorul cu 1. Prezentăm în continuare doar schema algoritmului de rezolvare, transcrierea în limbaj de programare rămânând cititorului ca exerciţiu.

Eficienţa algoritmului constă în faptul că nu sunt citite în memorie toate prenumele, de unde economia de memorie, iar lungimea maximă şi numărul de prenume având această lungime se realizează într-un singur pas, simultan cu citirea, fără a mai face parcurgeri suplimentare a tuturor prenumelor.

```
┌cat timp nu e sf. de fisier
│  citeste nume
│  l← lungime(nume)
│  ┌daca l>max atunci max←l; k←1
│  │  altfel ┌daca l=max atunci k←k+1
│  │         └■
└■ └■
```

67. Algoritmul este unul extrem de simplu, şi tocmai de aceea vom prezenta foarte pe scurt algoritmul de rezolvare.

```
--deschidere fişier
--citeste din fisier s
--citeste din fisier k
--închidere fisier
```

Varianta Pascal	*Varianta C/C++*
`for i:=1 to k do write(s[k]);`	`for (i=0;i<k;i++)cout<<s[k-1];`

68. *Varianta Pascal*

```
program ex;
var cuv:string;
    ok,i:integer;
    f:text;
begin
assign(f,'bac.txt'); reset(f);
readln(f,cuv); close(f);
i:=1;
while i<=length(cuv) do
 if pos(cuv[i],'aeiou')=0 then
    delete(cuv,i,1)
 else i:=i+1;
ok:=1;
for i:=1 to length(cuv)-1 do
  if cuv[i]>=cuv[i+1] then
    ok:=0;
if ok=1 then writeln('corect')
else  writeln('incorect');
readln;
end.
```

Varianta C/C++

```
#include <fstream.h>
#include <string.h>
#include <conio.h>

main()
{char cuv[100];
 ifstream f("bac.txt");
 f>>cuv; int i=0;
 while (i<strlen(cuv))
   if
(strchr("aeiou",cuv[i])==NULL)
    strcpy(cuv+i,cuv+i+1);
  else i++;
 int ok=1;
 for (i=0;i<strlen(cuv);i++)
   if (cuv[i]>=cuv[i+1]) ok=0;
if (ok) cout<<"corect";
 else cout<<"incorect";
 }
```

O altă idee de rezolvare: se copiază într-un al doilea şir de caracatere doar vocalele care apar în şirul citit. Programul va afişa “corect” numai dacă acest al doilea şir are elementele ordonate strict crescător.

69. *Varianta Pascal*

```
type sir=string[10];
var f:text;  s:sir;
    n:longint; k,i:byte;
    v:array[1..100] of longint;
function exista(n:longint):byte;
begin exista:=0;
  for i:=1 to k do
    if n=v[i] then exista:=1;
end;
procedure gen( var t:sir; i:byte);
var n,m:longint; e:integer;
begin  n:=length(t);
  if n>1 then begin
   if i=1 then delete(t,1,1)
          else delete(t,n,1);
```

Varianta C/C++

```
#include <fstream.h>
#include <string.h>
long v[100]; int k;
int exista(long m)
{ int i;
  for(i=1;i<=k;i++)
    if (m==v[i])return 1;
  return 0; }
void gen(char t[ ], int i)
{ int n=strlen(t); long m;
  if (n>1)
   {
    if (i==1) strcpy(t,t+1);
       else t[n-1]='\0';
    m=atol(t);
```

```
   val(t,m,e);
   if exista(m)=0 then begin
    inc(k); v[k]:=m; gen(t,i);end;
end;end;
Begin  assign (f,'NR.TXT'); rewrite(f);
  readln(n);  str(n,s); gen(s,1);
  str(n,s); gen(s,2);
  for i:=1 to k do  writeln(f,v[i]);
  close(f)
end.
```

```
    if(exista(m)==0)
      {++k; v[k]=m; gen(t,i);}  }}
void main()
{ ofstream f("NR.TXT");
  char s[9],t[9]; long n; cin>>n;
  ltoa(n,s,10); gen(s,1);
  ltoa(n,s,10); gen(s,2);
  for(int i=1;i<=k;i++)
     f<<v[i]<<endl;
f.close();
}
```

70. Implementăm functia **palindrom** care returnează **true/1** dacă numărul dat ca parametru are proprietatea cerută şi o utilizăm pentur a rezolva cerinţa din enunţ.

Varianta Pascal

```
var f:text;
    x:longint;
    am_afisat:boolean;
Function
palindrom(x:longint):boolean;
var y,r;
begin r:=0;y:=x;
while(x>0){r:=r*10+x mod 10;
           x:=x div 10;}
Palindrom:=(y=r)
end;
Begin
Assign(f,'bac.txt');
reset(f);
am_afisat:=false;
While not eof(f) do
begin
Read(f,x);
if palindrom(x) then
begin
 write(x,' ');
 am_afisat:=true
end
end;
if not am_afisat then  write(-1);
close(f)
end.
```

Varianta C/C++

```
# include <fstream.h>
ifstream f("bac.txt");
long x;
int am_afisat;
int palindrom(long x)
{ long y=x, r=0;
  while(x){ r=r*10+x%10;
            x=x/10;}
 return y==r;
}
void main()
{ while(! f.eof())
{ f>>x;
  if(palindrom(x))
        { cout<<x<<' ' ;
          am_afisat=1;
        }
}
if(! am_afisat)cout<<-1;
f.close();}
```

71. *Varianta Pascal*

```
type cuvant=string[20];
     mat=array[1..100] of
        cuvant;
var cuv:mat;
    n,i:integer;
    f,g:text;
procedure ordonare;
var i,j:integer;
    aux:cuvant;
```

Varianta C/C++

```
void ordonare
(char cuv[][20],int n);
void main()
{FILE f=fopen("CUVINTE.IN","r"),
    *g=fopen("CUVINTE.OUT","w");
/ fstream
/f("CUVINTE.IN",ios::in),
/g("CUVINTE.OUT",ios::out);
char cuv[100][20]; int n,i;
```

```
begin
for i:=1 to n do
   for j:= i+1 to n do
     if length(cuv[i])>
           length(cuv[j])
        then begin
          aux:=cuv[i];
          cuv[i]:=cuv[j];
          cuv[j]:=aux
       end
end;
begin
 assign(f,'CUVINTE.IN');
 assign(g,'CUVINTE.OUT');
 reset(f); rewrite(g);
 readln(f,n);
 for i:=1 to n do
    readln(f,cuv[i]);
 ordonare;
 for i:=1 to n do
     writeln(g,cuv[i]);
 close(f);
 close(g);
end.
```

```
fscanf(f,"%d",&n);
/f>>n;
for(i=0;i<n;i++)
   fscanf(f,"%s",cuv[i]);
   /f>>cuv[i];
ordonare(cuv,n);
for(i=0;i<n;i++)
   fprintf(g,"%s\n",cuv[i]);
/g<<cuv[i]<<endl;
 fclose(f); fclose(g);
/f.close(); g.close();
}
void ordonare
      (char cuv[][20],int n)
{int i,j; char aux[20];
for(i=0;i<n;i++)
  for(j=i+1;j<n;j++)
  if(strlen(cuv[i])>
         strlen(cuv[j]))
     {strcpy(aux,cuv[i]);
      strcpy(cuv[i],cuv[j]);
       strcpy(cuv[j],aux); }
}
```

72. *Varianta Pascal*

```
var f,g:text; s1,s2:string[15];
    i,j:integer;
begin assign(f,'X.TXT'); reset(f);
 assign(g,'Y.TXT'); reset(g);
 readln(f,s1); readln(g,s2);
 i:=1;
 j:=1;
 while (i<=7) and (j<=7) do
 if s1<s2 then begin  write(s1,' ');
          readln(f,s1); i:=i+1 end
    else begin write(s2,' ');
         readln(g,s2); j:=j+1 end;
 while i<=7 do begin  write(s1,' ');
          readln(f,s1); i:=i+1; end;
 while j<=7 do begin write(s2,' ');
          readln(g,s2); j:=j+1; end;
 close(f);
 close(g)
end.
```

Varianta C/C++

```
#include <fstream.h>
#include<string.h>
void main()
{char s1[16],s2[16];  int i,j;
 ifstream f("X.TXT"),g("Y.TXT");
 f>>s1; g>>s2; i=1; j=1;
 while (i<=7 && j<=7)
 if (strcmp(s1,s2)<0)
    {cout<<s1<<" ";f>>s1; i++;}
 else
     {cout<<s2<<" "; g>>s2;j++;}
  while (i<=7)
     {cout<<s1<<" ";f>>s1; i++;}
  while (j<=7)
     {cout<<s2<<" ";g>>s2; j++;}
  f.close();
  g.close();
}
```

73. *Varianta Pascal*

```
var f,g:text;s,cuv:string;L,k:integer;
begin
 assign(f,'cuvinte.in');
 reset(f);
 assign(g,'cuvinte.out');
 rewrite(g); read(L);
```

Varianta C/C++

```
void main()
{int L,k;
 char cuv[256];
 ifstream f("cuvinte.in");
 ofstream g("cuvinte.out");
 cin>>L;
```

```
while not eof(f) do begin
   readln(f,s);
   while length(s)<>0 do
    begin k:=pos(' ',s);
      cuv:=copy(s,1,k-1);delete(s,1,k);
      if k=0 then
         begin cuv:=s; s:=''; end;
      if length(cuv)=L then
         write(g,cuv,' ');
    end;
 end;
 close(f); close(g)
end.
```

```
while(!f.eof())
   {f>>cuv;
    if(strlen(cuv)==L)
       g<<cuv<<" ";
   }
 f.close();g.close();
}
```

74. *Varianta Pascal*

```
var f,g:text; i,j,v1,v2:integer;
begin assign(f,'A.TXT'); reset(f);
 assign(g,'B.TXT'); reset(g);
 readln(f,v1);readln(g,v2);
 i:=1;j:=1;
 while (i<=7) and (j<=7) do
 if v1>v2
   then begin write(v1,' ');
      readln(f,v1); i:=i+1  end
    else begin write(v2,' ');
      readln(g,v2); j:=j+1 end;
while i<=7 do begin write(v1,' ');
     readln(f,v1); i:=i+1; end
while j<=7 do begin write(v2,' ');
     readln(g,v2); j:=j+1; end;
close(f);
close(g)
end.
```

Varianta C/C++

```
#include <fstream.h>
#include<string.h>
void main()
{int i,j,v1,v2;
 ifstream f("A.TXT"), g("B.TXT");
 f>>v1; g>>v2;
 i=1; j=1;
 while (i<=7 && j<=7)
 if (v1>v2) {cout<<v1<<" ";
             f>>v1; i++;}
 else {cout<<v2<<" ";
       g>>v2;j++;}
  while (i<=7) {cout<<v1<<" ";
                f>>v1; i++;}
  while (j<=7) {cout<<v2<<" ";
                g>>v2; j++;}
  f.close(); g.close(); }
```

75. *Varianta Pascal*

```
var g:text;
s :array[1..5] of string[100];
i,j:integer;sw:boolean;
begin
assign(g,'BAC.TXT'); rewrite(g);
for i:=1 to 4 do readln(s[i]);
sw:=false;
for i:=1 to 4 do
 for j:=i+1 to 4 do
 if ((pos(s[i],s[j]))<>0)
or ((pos(s[j],s[i]))<>0)
  then
 begin writeln(g,s[i],',',s[j]);
  sw:=true;  end;
if (not sw)then
 writeln(g,'FARA SOLUTIE');
close(g)
end.
```

Varianta C/C++

```
#include<fstream.h>
#include<string.h>
void main()
{char s[5][100];
 int i,j,k;
 ofstream f("BAC.TXT");
 for(i=1;i<=4;i++)
       cin>>s[i];
 k=0;
 for(i=1; i<=4;i++)
  for(j=i+1;j<=4;j++)
  if ((strstr(s[i],s[j])) ||
           (strstr(s[j],s[i])))
     {g<<s[i]<<","<<s[j]<<endl;
      k++; }
 if (k==0)
    g<<"FARA SOLUTIE"; g.close();}
```

76. *Varianta Pascal*

```
var s:string[100]; i,j:integer;
    f:text; sterg:boolean;
begin assign(f,'DATE.TXT');
 rewrite(f);  readln(s);
 repeat
   sterg:=false; i:=1;
   while (i<length(s)) do
    begin
     if s[i]=s[i+1] then
       begin j:=i;
         while (s[i]=s[i+1])
          and (i<length(s))
                   do i:=i+1;
         delete(s,j,i-j+1);
         sterg:=true; break
        end;
   i:=i+1
  end
 until not sterg;
 writeln(f,s); close(f) end.
```

Varianta C/C++

```
#include<iostream.h>
#include<stdio.h>
#include<string.h>
#include<fstream.h>
void main()
{ofstream f("DATE.TXT");
 char s[100],ss[100];
 unsigned int i,j,sterg; cin>>s;
 do
  {sterg=0;
   for(i=0;i<strlen(s)-1;i++)
    if(s[i]==s[i+1])
     {j=i;
      while(i<strlen(s)-1&&
          s[i]==s[i+1]) i++;
      strcpy(s+j,s+i+1);
      sterg=1;break; }
  } while(sterg);
 f<<s; f.close();
}
```

77. *Varianta Pascal*

```
var f:text;s:string;
    i,nv,nc:integer;
function e_vocala(c:char):boolean;
begin
if pos(c,'aeiouAEIOU')<>0
     then e_vocala:=true
     else e-vocala:=false;
end;
begin assign(f,'bac.out');
 rewrite(g); readln(s);
 for i:=1 to length(s) do
    if e_vocala(s[i]) then
          inc(nv)
    else  inc(nc);
 if nv>=nc then begin
    for i:=1 to length(s) do
      if e_vocala(s[i]) then
          s[i]:=succ(s[i])
 end
 else for i:=1 to length(s) do
      if not e_vocala(s[i]) then
          s[i]:=pred(s[i]);
 writeln(f,s);  close(f); end.
```

Varianta C/C++

```
#include<iostream.h>
#include<stdio.h>
#include<string.h>
#include<fstream.h>
int e_vocala(char c)
{char v[]="aeiouAEIOU";
 return strchr(v,c)!=NULL;}
void main()
{char s[256]; cin>>s;
 int i,nv=0,nc=0;
 ofstream f(„bac.out");
 for(i=0;i<strlen(s);i++)
    if(e_vocala(s[i])) nv++;
    else nc++;
 if(nv>=nc)
   {for(i=0;i<strlen(s);i++)
    if(e_vocala(s[i])) s[i]++;}
 else
   for(i=0;i<strlen(s);i++)
     if(!e_vocala(s[i])) s[i]--;
 f<<s; f.close();}
```

78. *Varianta Pascal*

```
var f:text;
    n,i,j,k,gasit:integer;
    cuv:string[20];
begin assign(f,'fisier.txt');
reset(f); readln(f,n);k:=0;
```

Varianta C/C++

```
# include <iostream.h>
# include <stdio.h>
# include <string.h>
void main()
{ FILE *f=fopen("fisier.txt","r");
```

```
for i:=1 to n do  begin
  readln(f,cuv);gasit:=0;
  for j:=1 to length(cuv)-2 do
  if (cuv[j]='b') and
(cuv[j+1]='a')and
(cuv[j+2]='c') then gasit:=1;
  if gasit=1 then k:=k+1;
  end;
close(f); writeln(k); end.
```

```
int k=0,n;
 char cuv[21];
 fscanf(f,"%d",&n);
 for(int i=1;i<=n;i++)
  { fscanf(f,"%s",cuv);
   if (strstr(cuv,"bac")) k++;
  }
  cout<<k;
}
```

79. *Varianta Pascal*

```
var n,i,j,k,este:integer;
    cuv:string[10]; f:TEXT;
begin assign(f,'cuvinte.in');
reset(f);readln(f,n);
for i:=1 to n do
begin
 readln(f,cuv);este:=1;
 for j:=1 to length(cuv) do
  if
cuv[j]<>cuv[length(cuv)-j+1]
then este:=0;
 if este=1 then begin
   writeln(cuv); k:=k+1; end;
end;
if k=0 then write('NU'); end.
```

Varianta C/C++

```
# include <iostream.h>
# include <stdio.h>
# include <string.h>
void main()
{ FILE *f=fopen("cuvinte.in","r");
 int n,k=0,j,i,este; char cuv[11];
 fscanf(f,"%d",&n);
 for(i=1;i<=n;i++)
  {fscanf(f,"%s",cuv);este=1;
   for( j=0;j<=strlen(cuv)/2;j++)
    if
 (cuv[j]!=cuv[strlen(cuv)-j-1])
     este=0;
   if (este) {cout<<cuv<<endl;k++;}
  } if (!k) cout<<"NU";}
```

80. *Varianta Pascal*

```
type cuvant=string[30];
var c1,c2,c3:cuvant;
    linie:string[50];
    f:text; i,j:integer;
procedure ordonare(c1,c2,c3:cuvant);
 var s:cuvant;
 begin
 if (c1>c2) then begin
        s:=c1; c1:=c2; c2:=s end;
 if (c1>c3) then begin
        s:=c1; c1:=c3; c3:=s end;
 if (c2>c3) then begin
        s:=c2; c2:=c3; c3:=s end;
 writeln(c1,' ',c2,' ',c3)
end;
begin assign(f,'cuvinte.in');
reset(f);
while not eof(f) do
 begin  readln(f,linie);
 i:=pos(' ',linie);
 c1:=copy(linie,1,i);
 delete(linie,1,i);
 j:=pos(' ',linie);
 c2:=copy(linie,1,j);
 c3:=copy(linie,j+1,30);
 ordonare(c1,c2,c3);  end;
close(f) end.
```

Varianta C/C++

```
# include <iostream.h>
# include <string.h>
void ordonare (char c1[31],
       char c2[31],char c3[31])
{char s[31];
 if (strcmp(c1,c2)>0)
   {strcpy(s,c1); strcpy(c1,c2);
    strcpy(c2,s);}
 if (strcmp(c1,c3)>0)
   {strcpy(s,c1); strcpy(c1,c3);
    strcpy(c3,s);}
 if (strcmp(c2,c3)>0)
   {strcpy(s,c2); strcpy(c2,c3);
    strcpy(c3,s);}
 cout<<c1<<" "<<c2<<" "<<c3;
}
void main()
{  char c1[31],c2[31],c3[31];
 ifstream f("cuvinte.in");
 while(!feof(f))
 {
   fscanf(f,"%s %s %s",c1,c2,c3);
   cout<<endl;
   ordonare(c1,c2,c3);
 }
fclose(f); }
```

```
81. deschide fişier
    citeşte n (din fişier)
    pentru i←1,n execută citeşte pi.x,pi.y (din fişier)
    ■
    pentru i←1,n execută
    |   dacă (10≤pi.x) şi(pi.x≤20) atunci
    |   |   dacă (10≤pi.y) şi (pi.y≤20) atunci afişează (pi.x,pi.y)
    |   |   ■
    |   ■
    ■
   închide fişier
```

4.1.1.

1. c	2. a	3. d	4. b	5. b	6. a
7. a	8. a	9. d	10. d	11. a	12. a
13. a	14. a	15. a	16. c	17. a	18. c
19. c	20. a	21. a	22. d	23. d	24. d
25. a	26. d	27. d	28. a	29. a	30. d
31. c	32. a	33. a	34. c	35. d	

4.1.2.

1. **1)** D. **2)** orice valoare pentru **n** astfel încât **9** şi **n** sunt numere prime între ele. **3)** 3 valori **3**, **7** şi **9**

2. **1)** 5
2) Se modifica instrucţiunile: **i←3** ➔ **i←2**
i←i+2 ➔ **i←i+1**

3)

Varianta Pascal

```
function max (x,y:integer):integer;
begin if x>=y then max:=x
              else max:=y;
end;
```

Varianta C/C++

```
int max (int x, int y)
  { return x>=y?x:y;}
```

3. **1)** 900. **2)** k=4, n=6

3)

Varianta Pascal

```
function f(n:integer):integer;
  begin
    f:=n*(n+1) div 2
  end;
```

Varianta C/C++

```
int f(int n, int k)
{ return n*(n+1)/2;
}
```

4. **1)** 3 6 10
2) a≤b≤c
3)

Varianta Pascal

```
procedure  interschimb
  (var x,y:integer);
begin
  x:=x+y; y:=x-y; x:=x-y;
end;
```

Varianta C/C++

```
void interchimb( int &x, &y)
{   x=x+y; y=x-y; x=x-y; }
```

5. *Varianta Pascal*

```
var n :longint;
function prim(n:longint):boolean;
 var d:longint;
 begin
   prim:=true;
   if n<2 then prim:=false
   else
    for d:=2 to trunc(sqrt(n)) do
      if n mod d=0 then
        begin prim:=false; break end;
 end;
Begin readln(n);
   while (n>0)and(prim(n)=true) do
        n:=n div 10;
   if n>0 then writeln('NU')
          else writeln('DA')
end.
```

Varianta C/C++

```
#include <iostream.h>
int prim(long n)
{
  if (n<2)return 0;
  long d;
  for(d=2;d*d<=n;d++)
     if(n%d==0)return 0;
  return 1;
}
void main()
{
  long n; cin>>n;
  while(n&&prim(n))n=n/10;
  if(n) cout<<"NU";
  else cout<<"DA";
}
```

6. a) *Varianta Pascal*

```
procedure sub (var a,b:longint);
  begin
     b:=a mod 10+b*10;
     a:=a div 10
  end;
```

Varianta C/C++

```
void sub(long &a, long &b)
{
   b=a%10+b*10; a=a/10;
}
```

b) O soluţie posibilă se poate obţine prin construirea în variabila **m** a răsturnatului numărului **n**; iniţial **m** are valoarea **0**,.

```
m←0; p←n
┌cât timp p≠0 execută sub(p,m)
└■
┌dacă n≠m atunci scrie 'NU'
│          altfel scrie 'DA'
└■
```

7. a) *Varianta Pascal*

```
function nr(a,b:real):integer;
var x:integer;
begin
  if a>b then
      begin
       a:=a+b; b:=a-b;
       a:=a-b;
      end;
  x:=trunc(b)-trunc(a);
  if a=trunc(a)
         then nr:=x+1
         else nr:=x;
end;
```

Varianta C/C++

```
int nr( float a, float b)
{
  int x;
      if(a>b)
       { a=a+b; b=a-b; a=a-b;}
  x=floor(b)-floor(a);
  if(a==floor(a)) return x+1;
     else return x;
}
```

b) Numărul de numere cerut este rezultatul apelului **nr(10^(k-1)/a,(10^k-1)/a)**.

8. a)

Varianta Pascal

```
function S1(n:longint):byte;
procedure S2(var n:integer; k:byte);
```

Varianta C/C++

```
int S1( long n)
void S2(long &n, int k)
```

b)

```
citeşte n,a,b;
┌ dacă n>0
│   atunci  k←S1(n); m←n; S2(n,a); S2(m,b);
│            scrie k*3-S1(n)-S1(m)
│   altfel ┌ dacă a*b=0 atunci scrie 0
│         │             altfel scrie 1
│         └■
└■
```

9. a)

Varianta Pascal

```
function produs(a:longint):longint;
var p,i:longint;
begin p:=1; i:=0;
   while a>1 do begin
    if i=2 then i:=3
           else i:=i+2;
    if a mod i = 0 then
      begin p:=p*i;
        while a mod i=0 do
            a:=a div i;
      end;
    end;
   produs:=p; end;
```

Varianta C/C++

```
long produs( long a)
{ long p=1, i=0;
  while(a>1)
   {
    if (i==2)i=3;
        else i=i+2;
    if (a%i==0)
       { p=p*i;
         while(a%i==0)
            a=a/i;
       }
   }
  return p;   }
```

b) Programul pseudocod următor reprezintă o soluţie pentru rezolvarea cerinţei problemei:

```
citeşte a,b,c
pa←produs(a); pb←produs(a); pc←produs(c)
┌dacă pa=pb şi pa=pc atunci scrie 'DA'
│  altfel scrie 'NU'
└■
```

10. a)

Varianta Pascal

```
procedure sub(var n:longint; k:byte);
var m:longint; x,c:byte;
begin p:=0; z:=2;
 while n>0 do
  begin
   z:=z*10+n mod 10;
   n:=n div 10 end;
 while z>2 do
  begin c:=z mod 10; z:=z div 10;
   if (c=k) and (x=0)
        then x:=1
        else n:=n*10+c
  end  end;
```

Varianta C/C++

```
void sub( long &n, int k)
{ long z=2;
  int x=0,c;
  while(n>0)
    {z=z*10+n%10;n=n/10;}
  while(z>2)
  {
    c=z%10; z=z/10;
    if ((c==k)&&(x==0))
        x=1;
        else n=n*10+c;
} }
```

c) Programul pseudocod următor reprezintă o soluţie a problemei:

```
citeşte a,b; k←0; p←0
┌cât timp (k<=9) şi (a*b>0) execută
│ c←a; d←b;
│ sub(a,k);
│ sub(b,k);
│ k←k+1;
│ ┌dacă a≠c and b≠d atunci p←p+1
│ └■
└■
scrie p
```

11. Generalizăm problema pentru numere naturale **n** cu cel mult 9 cifre (care pot să conţină şi cifre nule; ultima intrucţiune while rezolvă aceste cazuri).

Varianta Pascal

```
var n,nr,m:longint; p,max:byte;
{cerinţa a)}
function putere(k:byte):longint;
procedure maxim(n:longint;
                var max,k:byte);
var s:string[10]; m,i:byte;
begin  str(n,s);
  m:=length(s); k:=m;
  for i:=m-1 downto 1 do
    if s[i]>s[k] then k:=i;
  max:=ord(s[k])-ord('0');
  k:=m-k;
end;
{cerinţa b)}
Begin readln(n); m:=n;
repeat
  maxim(n,max,p);
  nr:=nr*10+max;
  n:=n-max*putere(p);
until n=0;
while m>nr do nr:=nr*10;
writeln(nr);
end.
```

Varianta C/C++

```
#include <iostream.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>
//cerinţa a)
long putere(int k)
void maxim( long n, int &max,
            int &k)
{ char s[10],c,*p;
  ltoa(n,s,10); strrev(s);
  for (c='9';c>='0';c--)
  { p=strchr(s,c);
    if(p)
    {max=c-'0'; k=p-s;break;} }
}//cerinţa b)
void main(){ long n,nr=0,m;int p,max;
  cin>>n; m=n;
  do { maxim(n,max,p); nr=nr*10+max;
       n=n-max*putere(p);
     } while(n);
 while(m>nr)nr=nr*10;
 cout<<nr;
}
```

12. *Varianta Pascal*

```
var k,p:integer; n,b:longint;
{cerinţa a)}
procedure f(var n:longint;
       k:longint; var p:integer);
begin p:=0;
 while (n mod k =0)and(n>1)
  do begin n:=n div k; inc(p);end;
end;
{cerinţa b)}
Begin
 readln(n);
 k:=2;
 b:=1;
```

Varianta C/C++

```
#include <iostream.h>
//cerinţa a)
void f(long &n, long k, int &p)
{p=0;
 while((n%k==0)&&(n>1))
    {n/=k; p++;}
}
//cerinţa b)
void main()
{ int k=2,p;
  long n,b;
  cin>>n;
  b=1;
```

```
while n>1 do begin
   f(n,k,p);
   if p*(p mod 2)>0 then b:=b*k;
   if k=2 then k:=3
      else inc(k,2);
 end;
write(b)
end.
```

```
while(n>1)
 { f(n,k,p);
   if(p&&(p%2))b=b*k;
   if(k==2)k=3;
      else k+=2; }
cout<<b;

}
```

13. *Varianta Pascal*

```
var k,nr:integer;p,k1,k2:longint;
function prim(n:longint):byte;
  var d,p:longint;
  begin  p:=1;d:=2;
  if n<2 then p:=0
  else
   while (d<=sqrt(n))and(p=1) do
        if n mod d=0 then p:=0
        else d:=d+1;
  prim:=p;
end;
{cerinţa a)}
procedure f(n:longint; var
              k1,k2:longint);
begin k1:=n-1; k2:=n+1;
 while prim(k1)=0 do dec(k1);
 while prim(k2)=0 do inc(k2);
end;
{cerinţa b)}
Begin readln(k); p:=2; nr:=0;
 repeat
  p:=p+2; f(p,k1,k2);
  if k2=k1+2
    then begin write(p,' ');
         inc(nr); end;
  until nr=k;
end.
```

Varianta C/C++

```
#include <iostream.h>

int prim(long n)
{ long d;
  if(n<2)return 0;
  for(d=2;d*d<=n;d++)
   if(n%d==0)return 0;
  return 1;
}
//cerinţa a)
void f(long n,long &k1,long &k2)
{
 k1=n-1; k2=n+1;
 while (!prim(k1))k1--;
 while (!prim(k2))k2++;
}
//cerinţa b)
void main()
{ int k,nr=0; long p=2,k1,k2;
 cin>>k;
 do{ p=p+2; f(p,k1,k2);
     if (k2==k1+2)
        {cout<<p<<' '; nr++;}
   }while (nr<k);
}
```

O altă rezolvare se bazează pe următoarea observaţie: subprogramul **f** (cerut la a) funcionează corect şi pentru valori neprime ale lui **k** În aceste condiţii, **a** va fi cel mai mare divizor pentru care **f** ne returnează valoarea **2** (atunci când folosim respectivul divizor ca al doilea parametru de apel):

```
citeşte n
a←1
┌dacă n e pătrat perfect atunci a←n
│altfel
│┌pentru k←[n/2],2,-1
│|  apelează f(n,k,p)
│| ┌dacă p=2 atunci a←k; break
│| └■
│└■
└■
scrie a
```

14. Putem obţine divizorii primi din descompunerea numărului **n** în factori primi.

```
d1←2
┌cat timp n%d1≠0 executa d1←d1+1
└■
d2←d1; d←d1+1
┌cat timp n>1 executa
│   ┌daca n%d=0 atunci
│   │     d2←d
│   │     ┌cat_timp n%d=0 executa n←n/d
│   │     └■
│   └■
│   d←d+1
└■
```

Secvenţa anterioară se va scrie într-un subprogram cu antetul :

Varianta Pascal	**procedure nrdiv(n:integer;var d1,d2:integer);**
Varianta C/C++	**void nrdiv(int n, int&d1,int&d2);**

Rezolvarea cerinţei b):

```
citeşte a,b
┌pentru i←a,b
│  NRDIV(i,d1,d2)
│ ┌dacă 10≤d1 şi d2≤99 atunci
│ │ scrie i
│ └■
└■
```

15.

Varianta Pascal

```
var max,nr:integer; n:longint;c:byte;
    f:text;
{cerinţa a)}
procedure ncif(n:longint;var c:byte);
begin c:=1;
 while n>9  do
    begin n:=n div 10; inc(c);end;
end;
{cerinţa b)}
Begin
assign (f,'DATE.TXT'); reset(f);
while not eof(f) do begin
   read(f,n); ncif(n,c);
      if c>max then
        begin nr:=1; max:=c; end
      else
         if max=c then inc(nr); end;
writeln(nr) end.
```

Varianta C/C++

```
#include <fstream.h>
//cerinţa a)
void ncif(long n, int &c)
{c=1;
 while(n>9)
    {n/=10; c++;}
}
//cerinţa b)
void main()
{ ifstream f("DATE.TXT");
  int max=0,nr=0,c; long n;
 while(f>>n)
 { ncif(n,c);
   if(c>max){nr=1; max=c;}
      else if (max==c)nr++; }
cout<<nr;
}
```

16. **a)** Corectitudinea subprogramului cerut este evaluată prin:

- corectitudinea sintactică a antetului,

function panta(.....):byte; | **int panta(.....);**

-transferul corect prin parametri,

n:longint | **long a**

- calculul diferenţei maxime
- corectitudinea globală (declararea variabilelor locale necesare, respectarea structurii etc.)

Varianta Pascal

```
function panta(n:longint):byte;
var max,min:byte;
begin max:=0;min:=9;
repeat
 if n mod 10 > max then max:=n mod 10;
 if n mod 10 < min then  min:=n mod 10;
 n:=n div 10;
until n=0;
panta:=max-min
end;
```

Varianta C/C++

```
int panta (long n)
{
int max=0,min=9;
do{
if (n%10>max) max=n%10;
if (n%10<min) min=n%10;
n=n/10;
}while(n>0);
return max-min;
}
```

b) Pentru ca numărul n să îndeplinească cerinţa din enunţ este suficient să verificăm faptul că diferenţa dintre cea mai mare cifră şi cea mai mică cifră a numărului **k** (pe care o determinăm apelând subprogramul **panta**) este egală cu **nr_cifre-1** (**nr_cifre** fiind numărul de cifre ale lui **k**).
De exemplu, numărul **k=25436** are **5** cifre şi **panta(25436)=4**

```
citeşte k
n←panta(k);  nr_cifre=0;
┌ repetă
│ nr_cifre←nr_cifre+1
│ k←[k/10]
└până când k=0
┌dacă n+1=nr_cifre atunci
│   scrie DA ( pe ecran)
│ altfel
│   scrie NU (pe ecran)
└■
```

17. a) Singurele numere naturale care au un număr impar de divizori sunt pătratele perfecte; deci o metodă eficientă de rezolvare constă în a verifica dacă rădăcina pătrată a numărului este o valoare întreagă. Metodele bazate pe numărarea divizorilor sunt corecte dar nu sunt la fel de eficiente.
b) Se citesc din fişier cele **n** valori de pe a doua linie, se testează fiecare cu funcţia de la **a),** însumându-se şi numărându-se doar valorile care au un număr impar de divizori. Nu este necesar un vector pentru reţinerea numerelor de pe a doua linie; valorile necesare pentru calculul mediei se actualizează după fiecare valoare citită.

Varianta Pascal

```
var g:text; x,s,k,n,i:longint;
function f(n:integer):byte;
begin
 if sqrt(n)=trunc(sqrt(n))
  then f:=1 else f:=0
end;
begin
 assign(g,'BAC.TXT');reset(g);
 readln(g,n);
 for i := 1 to n do begin
   read(g,x);
   if f(x)=1 then begin
     s := s + x; k:=k+1
   end
 end;
 if k=0 then
    writeln('NU EXISTA')
 else writeln(s/k:5:2)
close(g)
end.
```

Varianta C

```
#include<stdio.h>
#include<math.h>
int f(int n)
{ int rad = floor(sqrt(n));
  if (n==rad*rad) return 1;
   else return 0;}
int main()
{FILE *g =fopen("BAC.TXT","r");
 int x; long s=0, k=0, n, i;
 fscanf(g,"%ld",&n);
 for (i=0;i<n;i++)
 {fscanf(g,"%d",&x);
   if (f(x)) {s+=x;k++;}
 }
 if (k==0)printf("NU EXISTA");
 else
   printf("%.2f",s/(k*1.));
g.close();
return 0;
}
```

18. Cerinţele se rezolvă prin algoritmi fundamentali aplicaţi pentru elementele unei matrice (însumare respectiv aflarea maximului); posibile implementări:

Varianta Pascal

```
type matr=array[0..9,0..9] of real;
var a:matr; i,j,n,p:byte;
    f:text; mx:real;
function f(a:matr, n,i:byte): real;
  var s:real; j:byte;
  begin  s:=0;
     for j:=0 to n-1 do
        s:=s+a[p,j];
     f:=s
  end;
begin  assign(g,'bac.out');
 rewrite(g); readln(n);
 for i := 1 to n do
  for j := 1 to n do
    readln(a[i,j]);
 mx := f(a,n,0);
 for i:= 1 to n-1 do
 if f(a,n,i)>mx then begin
   mx:=f(a,n,i);p:=i end;
 for j := 0 to n-1 do
     write(g,a[p,j],' ');
 close(g) end.
```

Varianta C

```
#include<stdio.h>
float f(float a[][10],int n,int p)
{float s=0; int j=0;
 for(;j<n;j++) s+=a[p][j];
 return s;
}
int main()
{float a[10][10],mx,aux;
 int n,i,j,p;
 FILE *g;
 g=fopen("BAC.OUT","w");
 scanf("%d",&n);
 for (i=0;i<n;i++)
   for (j=0;j<n;j++)
     scanf("%f",&a[i][j]);
 mx = f(a,n,0); p=0;
 for (i=1;i<n;i++)
 {aux = f(a,n,i);
  if (aux>mx){mx=aux; p=i;} }
 for (j=0;j<n;j++)
  fprintf(g,"%f ",a[p][j]);
 fclose(g); return 0;}
```

19. *Varianta Pascal*

```
var n:longint; s:real;
    fi,fo:text;
function sum(n:longint):real;
var ss:real; j:longint;
 begin  ss:=0;
  for j := 1 to n do
         ss:=ss+1/sqrt(j);
  sum:=ss
end;
begin
 assign(fo,'bac.out');
 rewrite(fo);
 assign(fi,'bac.in');reset(fi);
  readln(fi,n);
  s:=sum(n)-sqrt(2)*sum(n div 2);
  writeln(fo,s:5:3);
  close(fo); close(fi)
end.
```

Varianta C

```
#include<stdio.h>
#include<math.h>
float sum(long n)
{float s=0;
 for(long j=1;j<=n;j++)
    s=s+1/sqrt(j);
 return s;}
int main()
{long n; float s;
 FILE *fi, *fo;
 fi=fopen("bac.in","r");
 fo=fopen("bac.out","w");
 fscanf(fi,"%ld",&n);
 s=sum(n)-sqrt(2)*sum(n/2);
 fprintf(fo,"%.3f",s);
 fcloseall();
 return 0;
}
```

20. Vom presupune că **a<b** (dacă nu este aşa, interschimbăm cele două valori). O implementare a unei funcţii eficiente trebuie să folosească faptul că vectorul este sortat deci să aplice căutarea binară; se determină astfel două poziţii: **pi** care reprezintă poziţia primului element din vector mai mare sau egal cu **a** şi **pf** care reprezintă poziţia ultimului element din vector mai mic sau egal cu **b**. Rezultatul returnat de funcţie va fi **pf-pi+1** Determinarea valorii **pi** se poate face ca în secvenţa de mai jos, iar determinarea valorii **pf** este similară:

Varianta Pascal

```
function count(...):integer;
 var pi,pf,s,d,m:integer;
 begin
  s:= 1; d:= 2500;
  if a>x[d] then
    begin f:=0; exit end;
  repeat
    m:= (s+d) div 2;
    if a=x[m] then break;
    if a<x[m] then d:= m-1;
    if a>x[m] then s:= m+1;
  until s>d;
  if a<=x[m] then pi:=m
     else pi:=m+1;
  ...
 end;
```

Varianta C

```
int count(...)
{int pi,pf,s,d,m;
 s=0; d=2499;
 if (a>x[d]) return 0;
 do
 {m = (s+d)/2;
  if (a==x[m]) break;
  if (a<x[m]) dr = m-1;
  if (a>x[m]) st = m+1;
 } while (s<=d);
 if (a<=x[m]) pi=m
     else pi=m+1;
 ...
}
```

21. *Varianta Pascal*

```
var f,g:text; a,b:integer;  x:real;
function interval
  (x:real;a,b:integer):integer;
 var t:integer;
 begin
  if a>b then begin
     t:=a;a:=b;b:=t end;
  if (a<=x) and (x<=b) then
           interval :=1
     else  interval :=0
end;
begin
assign(f,'numere.in'); reset(f);
assign(g,'numere.out');rewrite(g);
readln(a,b);
 while not eof(f) do begin
  readln(f,x);
  if interval(x,a,b)=1 then
   write(g,x:5:2,' ')
 end;
 close(f); close(g) end.
```

Varianta C++

```
int interval(float x,int a,int b)
{ int x;
if (a>b) {x=a;a=b;b=x;}
 if (a<=x && x<=b) return 1;
return 0;
}
void main()
{int a,b;  float x;
 cin>>a>>b;
 ifstream f("numere.in");
 ofstream g("numere.out");
 while (f>>x)
 { if (interval(x,a,b)==1)
   g<<setprecision(2)<<x<<' ';
 }
 f.close();g.close();
}
```

22. *Varianta Free Pascal*
(se foloseşte numele funcţiei ca oricare altă variabilă, nefiind necesară o variabilă contor)

```
function vocal(s:string):byte;
var i;byte;
begin
  vocal:=0;
  for i:= 1 to length(s) do
    if pos(s[i],'aeiouy')>0 then
       inc(vocal);
end;
```

Varianta C/C++

```
int vocal(char s[])
{int ct=0,i;
 for (i=0;i<strlen(s);i++)
   if (strchr("aeiouy",s[i]))
     ct++;
  return ct;
}
```

23. Pentru fiecare linie calculăm suma şi comparăm elementul de pe diagonala principală a respectivei linii cu suma calculată, ţinând cont că în această sumă am adunat şi elementul de pe diagonala principală, pe care trebuie deci să îl scădem!

Varianta Pascal

```
type
   ma=array[1..20,1..20] of integer;
function
    matdiag (var m:ma;n:byte):byte;
var i,j,s;integer;
begin matdiag := 0;
 for i:= 1 to n do
 begin s:=0;
  for j := 1 to n do
    s:= s + m[i,j];
  if m[i,i]<s-m[i,i] then exit
 end;
 matdiag := 1; end;
```

Varianta C/C++

```
int matdiag(int m[][20],int n)
{int s,i,j;
 for (i=0;i<n;i++)
  {for(s=0,j=0;j<n;j++)
      s+=m[i][j];
   if (m[i][i]<s-m[i][i])
      return 0;
  }
  return 1;
}
```

24. *Varianta Pascal*

```
type vct=array[1..20] of integer;
function vect01(n:bye; v:vct):byte;
 var i,c0,c1;integer;
 begin c0:=0; c1:=0;
  for i:= 1 to n do begin
   if v[i]=0 then c0:=c0+1;
   if v[i]=1 then c1:=c1+1;
  end;
  if (n-1=c0)and(c1=1)
   then vect01:=1
   else vect01:=0;  end;
```

Varianta C/C++

```
int vect01(int n, int v[])
{int i,c0=0,c1=0;
 for (i=0;i<n;i++)
  { if (v[i]==0) c0++;
    if (v[i]==1) c1++;
  }
 if (c0==n-1 && c1==1)
      return 1;
 return 0;
}
```

25. **b)** Utilizând apeluri ale funcţiei definită la cerinţa a), se compară numărul de apariţii al fiecărui caracter din primul şir cu numărul de apariţii pe care le are acelaşi caracter în al doilea şir. Dacă şirurile au lungimi diferite, sigur există cel puţin un caracter care nu apare de acelaşi număr de ori în cele două şiruri (deci nu are rost să comparăm lungimile celor două şiruri)!

Varianta Pascal

```
var s1,s2:string[100];
    i,x:byte;
function nrap(s:string;c:char):integer;
...
begin  x:=1;
 readln(s1);readln(s2);
 for i:= 1 to length(s1) do
   if nrap(s1,s1[i]) <>
         nrap(s2,s1[i]) then
     begin x:= 0; break end;
 if x=1 then writeln('DA')
     writeln('NU');
end.
```

Varianta C++

```
#include<string.h>
#include<iostream.h>
int nrap(char s[], char c);
void main()
{char s1[101],s2[101];
 int i,x=1;
 cin>>s1; cin>>s2;
 for (i=0;i<strlen(s1);i++)
   if (nrap(s1,s1[i])!=
       nrap(s2,s1[i])
        {x=0; break;}
 if (x==1) cout<<"DA";
    else cout<<"NU";}
```

26. Vom da o soluţie care pleacă de la observaţia că numărul de zecimale care trebuiesc însumate este mic şi se bazează pe conversiile cu format de la un număr la un şir de caractere

(acestea permit reprezentarea unui număr real într-un şir de caractere cu un număr de zecimale pe care îl putem stabili noi). Trebuie să fim atenţi când însumăm elementele şirului de caractere, deoarece acestea nu conţin valoarea cifrelor ci codurile ASCII ale acestora!

Varianta Pascal

```
procedure sumazec(n:longint; c:byte;
                  var s:byte);
var sir:string[11];   i:byte;
begin  str(1/n:12:10,sir);  s:=0;
  for i := 3 to c+2 do
    s:= s + ord(sir[i])-ord('0');
end;
```

Varianta C

```
void sumazec(long n,int c,int &s)
{ char sir[13]; int i;
  sprintf(sir,"%.10f",1.0/n);
  s=0;
  for (i=2;i<=c+1;i++)
      s = s + sir[i]-'0';
}
```

27. a)

Varianta Pascal

```
type vct = array[1..100] of integer;
function
 pozamax(v:vct;n,p:integer):integer;
procedure schimb(var a,b : integer);
```

Varianta C++

```
int pozmax(int v[],int n, int p);

void schimb(int &a, int &b);
```

b) Folosind cele două subprograme, putem sorta elementele unui vector prin metoda sortării prin selecţie: în pasul `i` (`i=n,n-1,…,2`) determinăm poziţia maximului dintre primele `i` elemente din vector prin apelul funcţiei **`pozmax`** şi apoi interschimbăm maximul cu elementul din poziţia `i`. O posibilă implementare în pseudocod:

```
citeşte n
 pentru i = 1,n citeşte a[i]
┌pentru i = n,2,-1 execută p← pozmax(a,n,i); schimb(a[p],a[i])
└■
┌pentru i = 1,n scrie a[i]  (urmat de un spaţiu)
└■
```

28. a) Corectitudinea antetului cerut este evaluată prin:

- corectitudinea sintactică a antetului,

```
procedure divizor(....);  |   void divizor(...);
```

- transferul corect prin parametri,

```
n:longint;var d,p:longint |   long n, long &d, long &p
```

b) Este suficient să verificăm dacă cel mai mic divizor prim al lui n este **2** şi dacă puterea la care acesta apare în descompunerea numărului n este mai mare sau egală cu x.

```
citeşte n,x
divizor(n,d,p)
┌dacă d=2 şi x<=p atunci scrie DA
│altfel scrie NU
└■
```

29. a) Corectitudinea antetului cerut este evaluată prin:

- corectitudinea sintactică a antetului,

```
function maxim(....):real;          |   float maxim(...);
```

- transferul corect prin parametri,

```
type vector=array[1..100]of real;

var V:vector;n:byte                 |   float V[101],int &n
```

b)

```
citeste n
┌pentru i←1,n execută citeste Vi
└■
┌cat timp n>0 execută x←maxim(V,n); scrie x
└■
```

30.

Varianta Pascal

```
type vect=array[1..50] of integer;
var a: vect;n,i,nr:byte;
function f(a:vect;n,x:byte):byte;
 var i,k:byte;
 begin
 k:=0;
 for i:=1 to n do
   if x>a[i] then inc(k);
 f:=k;
 end;
begin
 readln(n);
 for i:=1 to n do read(a[i]);
 nr:=f(a,n,1)-f(a,n,0);
 writeln(n-nr);
end.
```

Varianta C/C++

```
#include <iostream.h>

int f(int a[50], int n, int x)
{ int k=0,i;
  for (i=0;i<n;i++) if(x>a[i])k++;
  return k;

}
void main()
{int a[50],n,i,nr;
for(cin>>n,i=0;i<n;cin>>a[i],i++);
 nr=f(a,n,1)-f(a,n,0);
 cout<<n-nr;
}
```

31.

a)

Varianta Pascal

```
procedure inter(var a,b:real)
begin
 a:=a+b;
 b:=a-b;
 a:=a-b;
end;
```

Varianta C/C++

```
void inter(float &a, float&b)
{ a=a+b;b=a-b;a=a-b;}
```

b)

Varianta Pascal

```
type
vector=array[1..99] of real;
Procedure circ(v:vector;n:integer)
Var i:byte;
begin
for i:=1 to n-1 do
          inter(v[i],v[i+1])
end;
```

Varianta C/C++

```
void circ(int v[99],int n)
{
  for(int i=0;i<n-1;i++)
         inter(v[i],v[i+1]);
}
```

32.

a)

Varianta Pascal

```
function
majuscula(var c:char):integer;
```

Varianta C/C++

```
int majuscule(char &c)
```

b)

Varianta Pascal

```
var f,g:text;
    c:char;
begin
assign(f,'bac.in');
assign(g,'bac.out');
reset(f);
rewrite(g);
while not eof(f) do
 begin
read(f,c);
if majuscule(c)=1 then
                write(g,c,'!')
          else write(g,c);
close(f);
close(g)
end.
```

Varianta C/C++

```
# include <fstream.h>
# include <string.h>
ifstream f("bac.in");
ofstream g("bac.out");
void main()
{ char c;
  while(f>>c)
 { if(majuscula(c))g<<c<<'!';
    else g<<c; }
  f.close();
  g.close();
}
```

33. *Varianta Pascal*

```
type vect=array[1..50] of integer;
var a: vect;n,i,nr:byte;
{cerinţa a)}
function f(a:vect;n:byte):byte;
 var i,k:byte;
 begin i:=1; f:=0;
 if a[1]<a[2] then begin
    while (a[i]<a[i+1])and(i<n) do
       i:=i+1;
    if i=n then f:=1;
   end
  else begin
    while (a[i]>a[i+1])and(i<n) do
         i:=i+1;
     if i=n then f:=2;
   end;
 end;
{cerinţa b)}
begin readln(n);
 for i:=1 to n do read(a[i]);
 i:=2; nr:=f(a,2);
 if nr=0 then i:=1
 else
 while (f(a,i+1)=nr)and(i<n) do
     inc(i);
 writeln(i);
end.
```

Varianta C/C++

```
#include <iostream.h>
//cerinţa a)
int f(int a[51], int n)
{ int i=1,k=0;
  if(a[1]<a[2])
   { while((a[i]<a[i+1])&&(i<n))
         i++;
     if(i==n) k=1; }
  else
   { while((a[i]>a[i+1])&&(i<n))
         i++;
     if(i==n) k=2; }
  return k; }

//cerinţa b)
void main()


{int a[50],n,i=2,nr; cin>>n;
 for(i=1;i<=n;cin>>a[i],i++);
 nr=f(a,2);
 if(nr==0)i--;
  else
  while((f(a,i+1)==nr)&&(i<n))i++;
 cout<<i;
}
```

34. *Varianta Pascal*

```
type vect=array[1..50] of integer;
 matrice=array[1..10,1..10] of integer;
  procedure vector(n,m:integer;
     var a:matrice;var v:vect);
 var i,j,k:integer;
 begin  k:=0;
  for j:=1 to m do
   for i:=n downto 1 do
    begin v[k]:=a[i,j];k:=k+1;
    end;
 end;
```

Varianta C/C++

```
void vector(int n,int m,
       int a[][30],int v[])
{
  int k=0,i,j;
  for(j=0;j<m;j++)
  for(i=n-1;i>=0;i--)
    v[k++]=a[i][j];
}
```

35. *Varianta Pascal*

```
{cerinţa a)}
procedure trans( var c:char);
{cerinţa b)}
var s:string[100]; i:byte;
Begin  readln(s);
 for i:=1 to ord(s[0]) do
      trans(s[i]);
 writeln(s);
end.
```

Varianta C/C++

```
#include <stdio.h>
#include <iostream.h>
// cerinţa a)
void trans( char &c)
// cerinţa b)
void main() { char s[101]; gets(s);
  int i,n=strlen(s);
  for(i=0; i<n; i++ )trans(s[i]);
  cout<<s;
}
```

36. *Varianta Pascal*

```
type sir=string[50];
var s:array[1..50] of sir;
    n,i:byte;
{cerinţa a)}
procedure f(s:sir;k:byte);
var v:sir;
begin
 if length(s)<k
   then writeln('nu este posibil')
   else   begin
            v:=copy(s,1,k);
            writeln(v)
          end;
end;
{cerinţa b)}
Begin readln(n);
for i:=1 to n do readln(s[i]);
for i:=1 to n do f(s[i],i);
end.
```

Varianta C/C++

```
#include <iostream.h>
#include <string.h>
// cerinţa a)
void f(char s[51], int k)
{ char v[51];
  if(strlen(s)<k)
    cout<<"nu este posibil"<<endl;
  else { strncpy(v,s,k);v[k]='\0';
         cout<<v<<endl; }
}
 // cerinţa b)
void main()
{ char s[51][51]; int i,n;
  cin>>n;
  for(i=1;i<=n;i++)
  { cin.get(); cin.get(s[i],51); }
  for(i=1; i<=n; i++) f(s[i],i);}
```

37. *Varianta Pascal*

```
type sir=string[100];
var f:text;
    s,t:sir;
{cerinţa a)}
procedure elimin(s:sir;var t:sir);
var v:string; i,p:byte;
begin v:='aeiou';
  t:=copy(s,1,length(s));
  for i:=1 to 5 do
   begin
       p:=pos(v[i],t);
    while p>0 do
     begin
     delete(t,p,1);
     p:=pos(v[i],t);
     end;
   end;
end;
{cerinţa b)}
Begin
assign (f,'bac.txt');
reset(f);
while not eof(f) do
   begin
    readln(f,s);
    elimin(s,t);
    writeln(s,' ',t);
   end;
close(f);
end.
```

Varianta C/C++

```
#include <fstream.h>
#include <string.h>
// cerinţa a)
void elimin(char s[101], char t[101])
{ char v[]="aeiou", *p;
  int i; strcpy(t,s);
  for(i=0;i<=4;i++)
  { p=strchr(t,v[i]);
    while(p)
    { strcpy(p,p+1);
      p=strchr(p,v[i]); }  }
}
// cerinţa b)
void main()
{ ifstream f("bac.txt");
  char s[101], t[101];
  while(!f.eof())
  { f>>s;
    elimin(s,t);
    cout<<s<<" "<<t<<endl;  }
f.close();}
```

38. *Varianta Pascal*

```
var s,t:string; i:byte;
{cerinţa a)}

function sub(var s:string;c:char):byte;
 var k,p:byte;
 begin p:=pos(c,s); k:=0;
   while p>0 do begin
      inc(k); delete(s,p,1);
      p:=pos(c,s)
     end;
   sub:=k;
 end;
{cerinţa b)}
Begin readln(s);
      readln(t);
   for i:=1 to length(s) do
     if sub(t,s[i])>0 then
        write(s[i],' ');
end.
```

Varianta C/C++

```
#include <string.h>
#include <iostream.h>
#include <string.h>
//cerinţa a)
int sub( char s[201], char c)
{ int k=0; char *p;
  p=strchr(s,c); k=0;
   while (p)
    { k++; strcpy(p,p+1);
      p=strchr(s,c);}
  return k;}
//cerinţa b)
void main()
{ char s[201], t[201]; int i;
  cin.get(s,200); cin.get();
  cin.get(t,200);
  for(i=0;i<strlen(s);i++)
   if (sub(t,s[i]))
      cout<<s[i]<<' '; }
```

39. *Varianta Pascal*

```
var x,y:string[127];
    n,i,nr:integer; f:text;
{cerinţa a)}
function ordalfa(a:string):byte;
var i,n:byte;
begin n:=length(a); i:=1;
   while (a[i]<=a[i+1])and(i<n) do
      i:=i+1;
   if i=n then ordalfa:=1
      else ordalfa:=0;
end;
{cerinţa b)}
Begin assign(f,'bac.txt'); reset(f);
 readln(f,n); readln(f,x);
 for i:=2 to n do
 begin readln(f,y);
  if ordalfa(x)*ordalfa(y)=1 then
   if (x[length(x)]<=y[1])or
      (y[length(y)]<=x[1])
         then  inc(nr);
  x:=y;
 end;
writeln(nr); end.
```

Varianta C/C++

```
#include<fstream.h>
#include <string.h>
//cerinţa a)
int ordalfa(char a[256])
{ int i=0, n=strlen(a)-1;
  while ((a[i]<=a[i+1])&&(i<n))
       i++;
  return (i==n);
}
//cerinţa b)
void main()
{ifstream f("bac.txt");
 char x[128],y[128];
 int n,i,nr=0;
  f>>n>>x;
 for(i=2;i<=n;i++)
{f>>y;
 if (ordalfa(x)*ordalfa(y))
  if((x[strlen(x)-1]<=y[0])||
     (y[strlen(y)-1]<=x[0]))
      nr++;
  strcpy(x,y);
} cout<<nr; }
```

40. a) *Varianta Pascal*

```
function max_diviz(a,b:longint):byte;
var i:integer;
begin max_diviz:=1;i:=99;
while (i>=10) do
if(a mod i=0) and (b mod i=0)
then  begin max_diviz:=i;i:=0;end;
 else i:=i-1; end;
```

Varianta C/C++

```
int max_diviz(long a,long b)
{ int i;
 for(i=99;i>=10;i--)
  if(a%i==0 && b%i==0)
     return i;
 return 1;
}
```

```
b) citeşte a
  divizor← max_diviz(a,a);
  ┌dacă divizor≠1 atunci scrie divizor
  │              altfel scrie mesaj
  └■
```

41. **a)** Descriem în pseudocod funcţia care întoarce ca rezultat valoarea **1** dacă numărul **n** transmis ca parametru are toate cifrele pare şi **0** în caz contrar.

```
     ┌întreg cifre_pare(întreg n)
     │  ┌cât timp n>0 execută
     │  │ ┌dacă n%2≠0 atunci întoarce 0
     │  │ └■
     │  │  n=[n/10]
     │  └■
     │  întoarce 1
     └■
b)   ┌pentru i←2,10000,2 execută
     │ ┌dacă cifre_pare(i)=1 atunci scrie i
     │ └■
     └■
```

42. Se construieşte funcţia cmmdc care returnează cel mai mare divizor comun al celor două numere **a** şi **b**, primite ca parametrii. Utilizând formula matematică de calcul al celui mai mare multiplu comun al numerelor **a** şi **b** **(cmmmc=(a*b)/cmmdc)**, putem apoi genera **m-1** multiplii comuni ai celor două numere **a** şi **b**.

```
citeşte a,b,m
d←cmmdc(a,b);
cmmmc←(a*b)/d
┌pentru i←1,m execută scrie i*cmmmc
└■
```

43. Vom folosi un algoritm iterativ, în care la fiecare pas vom reţine în trei variabile valorile a trei termeni consecutivi ai şirului, într-o a patra variabilă vom determina următorul termen. Algoritmul este eficient d.p.d.v. al spaţiului de memorie pentru că nu memorează într-un vector termenii şirului, şi este deasemenea eficient d.p.d.v. al timpului de executare pentru comparativ cu algoritmul recursiv, deoarece în cazul acestuia din urmă, unii termeni ai şirului se calculează de mai multe ori.

Algoritmul în pseudocod este următorul:

```
┌dacă n≤3 atunci returnează n
|  altfel   a←1; b←2; c←3
|           ┌pentru i←4,n execută
|           |  d←(c*b+a+b)%n; a←b; b←c; c←d
|           └■
|           returnează d
└■
```

44. *Varianta Pascal*

```
...
function sub(nr:integer):integer;
var i,a,b:integer;
begin a:=0; b:=0;
if nr=0 then a:=1;
while nr>0 do begin
  if nr mod 10 mod 2=0 then
    a:=a+1
  else  b:=b+1;
  nr:=nr div 0; end;
if a=b then sub:=1
       else sub:=0;
end;
begin
 ...
 assign(f,'bac.txt');
 reset(f); read(f,n);
 for i:=1 to n do read(f,v[i]);
 close(f); max:=0; k:=0;
 for i:=1 to n do
  begin
   if sub(v[i])=1 then k:=k+1
                  else k:=0;
   if k>max then
   begin max:=k; sf:=i; end;
 end;
for i:=sf-max+1 to sf do
  write(v[i],' ,);
end.
```

Varianta C/C++

```
...
int sub(int nr)
{  int i,a,b;
   a=b=0;
   if (a==0) return 0;
   while (nr>0)
   { if (nr%10%2==0) a++;
     else b++;
     nr=nr/10;  }
   if (a==b) return 1;
   else return 0;}
main()
{
....
ifstream f("bac.txt");
f>>n;
for (i=1;i<=n;i++) f>>v[i];
f.close(); max=k=0;
for (i=1;i<=n;i++)
 {if (sub(v[i])) k++;
  else k=0;
  if (k>max) { max=k;
               sf=i;
             }
  }
for (i=sf-max+1;i<=sf;i++)
  cout<<v[i]<<" ";
}
```

45. *Varianta Pascal*

a)

```
function numar(a:word):word;
```

b)

```
var k:word;
begin read(k);
if numar(k)=2
     then write('DA')
     else write ('NU')
end.
```

Varianta C/C++

```
unsigned numar(unsigned a)
```

```
unsigned k;
void main()
{ cin>>k;
  if(numar(k)==2) cout<<"DA";
              else cout<<"NU";
}
```

46. *Varianta Pascal*

a)

```
function fibo(n:word):word;
```

b)

```
var n,f:word;
Begin
readln(n);
while n>0 do
begin
  f=fibo(n); write(f, ' ');
  n:=n-f end
end.
```

Varianta C/C++

```
unsigned fibo(unsigned n)
```

```
# include <iostream.h>
unsigned n,f;
void main(){
cin>>n;
while(n)
 { f=fibo(n);
   cout<<f<<' '; n=n-f; }
}
```

47. a) Pentru a determina în **y** cel mai mare divizor al lui **a** (diferit de **a**), căutăm cel mai mic divizor **x** $\in [2;\sqrt{a}]$ şi obţinem **y=a/x**. Un astfel de număr **x** nu există atunci când a este prim. În acest caz **y=1**. Valoarea numărul x se determină cu ajutorul unui algoritm de complexitate $O([\sqrt{a}])$. Se observă că **2** este singura valoare pară ce va fi testată pentru **x**.

b)

Varianta Pascal

```
function divizor(a:longint):longint;
var x,y:longint;
begin
if a mod 2=0 then y=a div 2
else begin x:=3;
     while (x*x<=a) and(a mod x>0) do x:=x+2;
     if x*x<=a then y:=a div x
               else y:=1
     end;
divizor:=y end;
```

Varianta C/C++

```
long divizor(long a)
{ long x;
if(a%2==0)return a/2;
for(x=3;x*x<=a;x=x+2)
 if(a%x==0) return a/x;
return 1;
}
```

c) Este suficient să verificăm dacă **divizor(n)=1.**

48. a) Corectitudinea sintactică a antetului subprogramului cerut este evaluată prin:

Pascal:
```
type matrice=array[1..10,1..10]of real;
function maxim(A:matrice;i1,i2:byte): real; |
```

C/C++:
```
float maxim(float A[10][10], int i1, int i2);
```

b) Pentru a calcula diferenţa maximă dintre două elemente ale matricei **A** aflate pe linii **diferite**, trebuie să apelăm funcţia **maxim** definită la punctul a) pentru toate perechile de linii **{(i,j)/i≠j}** din matricea **A**.

```
┌pentru i←1,10 execută
│  ┌pentru j←1,10 execută citeste Aij
│  └■
└■
difmax←maxim(9,10)
┌pentru i←1,8 execută
│ ┌pentru j←i+1,9 execută
│ │  x←maxim(i,j)
│ │  ┌dacă x>difmax atunci difmax←x
│ │  └■
│ └■
└■
scrie difmax
```

Obs.: In limbajul C/C++ indicii variază în intervalul **[0;9]**.

49. a)
```
întreg f(întreg n)
   ┌dacă n=1 atunci întoarce 0
   │  altfel
   │    t1←1;t2←1;
   │   ┌cât timp t2<n execută
   │   │  t3←t1+t2;t1←t2;t2←t3;  |
   │   └■
   │   întoarce t1
   └■
```

b) descriem în pseudocod algoritmul pentru rezolvarea problemei

```
citeşte dim
┌pentru i←1,dim execută citeşte x[i]
└■
┌pentru i←1,dim execută
│  x[i]←f(x[i]); scrie x[i]
└■
```

50. a)
```
întreg s1(întreg sir, întreg dim, întreg p, întreg q)
 max← sir[p]
 ┌pentru i←p+1,q execută
 │  ┌dacă sir[i]>max atunci  max←sir[i]
 │  └■
 └■
 întoarce max
```

Descrierea subprogramului **s2** este similară cu cea pentru **s1**.

b) prezentăm în continuare o variantă de rezolvare a problemei, descrisă în pseudocod

```
citeşte dima,a; citeşte dimb,b
Min=s2(b,dimb,1,dimb)
Max=s1(a,dima,1,dima)
┌dacă Max<Min atunci scrie "Da" altfel scrie "NU"
└■
```

51.

Varianta Pascal

```
type numar=record
            nr:longint;  s:integer;
           end;
var x:array[1..100] of numar;
    n,i,j:integer; aux:numar;
function sc(n:longint):integer;
var s:integer;
begin s:=0;
 while n<>0 do begin
   s:=s+n mod 10; n:= n div 10 end;
 sc:=s;
end;
procedure citire;
var i:integer;
begin read(n);
 for i:=1 to n do
    begin
     read(x[i].nr);
     x[i].s:=sc(x[i].nr) end
end;
begin citire;
  for i:=1 to n-1 do
    for j:=i+1 to n do
      if x[i].s<x[j].s then begin
            aux:=x[i]; x[i]:=x[j];
            x[j]:=aux
           end;
 for i:=1 to n do
    write(x[i].nr,' ');
end.
```

Varianta C/C++

```
struct numar {long nr;
              int s;};
numar x[101];
int n;
int sc(long n)
{int s=0;
 while(n)
  { s=s+n%10; n=n/10;}
 return s;
}
void citire()
{int i;
 cin>>n;
 for(i=0;i<n;i++)
   {cin>>x[i].nr;
    x[i].s=sc(x[i].nr);}
}
void main()
{int i,j; numar aux;
 citire();
 for(i=0;i<n;i++)
  for(j=i+1;j<n;j++)
    if(x[i].s<x[j].s)
      {aux=x[i]; x[i]=x[j];
       x[j]=aux;}
 for(i=0;i<n;i++)
   cout<<x[i].nr<" ";
}
```

52.

```
┌întreg cmmdc (intreg a, întreg b)
│  întreg r
│  ┌repetă
│  │  r←a%b; a←b; b←r
│  └■până când r=0
└■ întoarce a
citeşte m,n
┌pentru i←1,m execută
│  ┌pentru j←1,n execută a_ij←cmmdc(i,j)
│  └■
└■
afişare matricea a
```

53.

Varianta Pascal

```
var a:aray[1..100][1..100] of byte;
n,i,j:integer;

function prim(nr:longint):boolean;
. . .
begin read(n);
for i:=1 to n do
  for j:=1 to n do
   if prim(i*i+j*j) then a[i,j]:=1
   else if (i+j) mod 2 = 0
        then a[i,j]:=2
        else a[i,j]:=0;
. . .
end.
```

Varianta C/C++

```
. . .
int prim(long n);
. . .
void main()
{unsigned a[101][101];
 int n,i,j; cin>>n;
 for(i=1;i<=100;i++)
   for(j=1;j<=100;j++)
     if(prim(i*i+j*j))a[i][j]=1;
     else if((i+j)%2==0)
            a[i][j]=2;
          else a[i][j]=0;
. . .
}
```

54. a) Subprogramul **elimin** va construi şirul **s** literă cu literă, luând în considerare toate literele alfabetului care aparţin cel puţin unuia dintre şirurile **s1** sau **s2**.

Varianta Pascal

```
procedure elimin
(s1,s2:string; var s:string);
var c:char;
begin
s:='';
for c:='a' to 'z' do
 if pos(c,s1)+pos(c,s2)<>0
       then  s:=s+c;
end;
```

Varianta C/C++

```
void elimin ( char s1[], char s2[],
              char s[])
{char c; int i=0;
 strcpy(s,"");
 for(c='a';c<='z';c++)
   if(strchr(s1,c)||strchr(s2,c))
                    s[i++]=c;
 s[i]=NULL;
}
```

b) Vom citi primul cuvânt în şirul **s1**, apoi pentru fiecare din cuvintele următoare vom citi cuvântul în şirul **s2**, apelăm funcţia **elimin** care construieşte şirul **s** cu toate literele ce apar cel puţin în unul din cele două şiruri şi apoi copiază rezultatul în **s1**

Varianta Pascal

```
var n,i:integer;
    s,s1,s2:string;
begin
 readln(n); readln(s1);
 for i:=2 to n do begin
    readln(s2);
    elimin(s1,s2,s);  s1:=s;
 end;
writeln(s1); end.
```

Varianta C/C++

```
int n,i;
char s[251],s1[251],s2[251];
void main()
{cin>>n; cin.get(); cin>>s1;
 for(i=2;i<=n;i++)
   {cin>>s2; elimin(s1,s2,s);
    strcpy(s1,s);}
 cout<<s1;
}
```

55. a) Corectitudinea subprogramului cerut este evaluată prin:

Varianta Pascal	*Varianta C/C++*

`type sir=string[100];`
`function aparitii(s:sir;x:char):byte;`
`var i,nr: byte;`
`begin nr:=0;`
`for i:=1 to length(s) do`
`if s[i]=x then nr:=nr+1;`
`aparitii:=nr end;`
	`int aparitii ( char s[101],`
	`char x)`
	`{ int nr=0,i;`
	`for(i=0;i<strlen(s);i++)`
	`if(s[i]==x) nr++;`
	`return nr;`
	`}`

b) Dacă şirurile `s1` şi `s2` au aceeaşi lungime şi nu sunt identice parcurgem unul dintre şiruri şi verificăm dacă fiecare literă apare de acelaşi număr de ori în ambele şiruri. S-a notat cu `li` indicele primului caracter din şirul `s1`.

```
citeste s1,s2; ok←1
┌dacă lung(s1)≠lung(s2)sau s1=s2 atunci scrie NU SUNT ANAGRAME
│ altfel
│    ┌ i←li
│    │ ┌cat_timp i<li+lung(s1) şi ok=1 execută
│    │ │ ┌dacă aparitii(s1,s1[i])≠ aparitii(s2,s1[i])atunci ok←0
│    │ │ │  altfel i←i+1
│    │ │ └■
│    │ └■
│    │ ┌dacă ok=1 atunci scrie ANAGRAME
│    │ │           altfel scrie NU SUNT ANAGRAME
│    │ └■
│    └■
└■
```

4.2.1.

1. b	2. b	3. a	4. d	5. d	6. b
7. d	8. c	9. b	10. a	11. c	12. c
13. d	14. c	15. a	16. a	17. a	18. b
19. c	20. c	21. d	22. d	23. a	24. a
25. d	26. a	27. b	28. b	29. d	30. a
31. d	32. a	33. b	34. b	35. a	36. a
37. a	38. b	39. b	40. d	41. b	42. c
43. c	44. c	45. d	46. a	47. b	48. a
49. a	50. d	51. d	52. b	53. c	54. c
55. a	56. c	57. b	58. b	59. b	60. a
61. c	62. c	63. b	64. d	65. b	66. d
67. c	68. c	69. b	70. a	71. d	

4.2.2.

1. **1)** **2456** şi **357**

2) Orice număr format din trei cifre impare.

4) O soluţie posibilă pentru descrierea în pseudocod a funcţiei recursive este:

```
întreg a
┌subprogram sub(întreg n)
│ ┌dacă (n%2≠0) atunci a←a*10+n%10; n←n/10; sub(n)
│ │             altfel scrie n,a
│ └■
└■
```

2. *Varianta Pascal*

```
procedure cif(n:longint);
begin
if n>0 then begin cif(n div 10);
                  if n mod 2>0 then
                  write(n mod 10); end;
end;
```

Varianta C/C++

```
void cif(long n)
{ if (n)
    { cif(n/10);
      if(n%2) cout<<n%10;}
}
```

3. Se observă că relaţia de recurenţă necesară este următoarea

sub(n) = maxim(sub([n/10]), n%10) dacă n≥10

şi sub(n) = n dacă n≤ 9

Aşadar subprogramul sub este următorul

Varianta Pascal

```
procedure sub(n:integer; var
max:integer)
begin
  if n<=9 then max:=n
  else
   begin sub(n div 10,max);
     if n mod 10>max then
        max:=n mod 10; end;
end;
```

Varianta C/C++

```
void sub(int n,int &max)
{ if (n<=9) max=n;
  else
  {  sub(n/10,max);
     if (n%10>max)
       max=n%10;
  }
}
```

Pentru a afla de câte ori apare cifra maximă într-un număr vom compara pe rând fiecare cifră a numărului, obţinute prin împărţire la 10, cu cifra maximă.

```
citeşte n; sub(n,max); k←0
┌cat timp n>0 executa
│  ┌daca n%10=max atunci k←k+1
│  └■
│  n←[n/10]
└■
scrie k
```

4. Utilizarea unei metode recursive este sugerată de faptul că numerele se scriu în ordinea inversă citirii (o altă metodă ar putea memora într-un tablou unidimensional numerele din vector, scrierea în cel de-al doilea fişier făcându-se prin parcurgerea vectorului în sens invers).

Varianta Pascal

```
Var f,g:text;
procedure cs;
var x:integer;
begin
if not eof(f) then  begin
   read(f,x); cs; write(g,x,' ')
                  end
end;
Begin assign(f,'date.in');reset(f);
assign(g,'date.out');rewrite(g);
cs;
close(f);close(g) end.
```

Varianta C/C++

```
# include <stdio.h>
FILE *f, *g;
void cs(){
int x;
if(!feof(f)){fscanf(f,"%d",&x);
             cs();
             fprintf(g,"%d ",x);
             }
}
void main()
{f = fopen("date.in","r");
 g = fopen("date.out","w");
cs(); fcloseall();}
```

5. Utilizarea unei metode recursive este sugerată de faptul că numerele se scriu în ordinea inversă citirii (o altă metodă ar putea memora într-un tablou unidimensional numerele din vector, scrierea în cel de-al doilea fişier făcându-se prin parcurgerea vectorului în sens invers).

Varianta Pascal

```
Var f,g:text;
procedure cs;
var x:integer;
begin
if not eof(f) do  begin
   read(f,x); cs; write(g,x,' ')
                  end
end;
Begin assign(f,'date.in');reset(f);
assign(g,'date.out');rewrite(g);
cs;
close(f);close(g) end.
```

Varianta C/C++

```
# include <fstream.h>
ifstream f("date.in");
ofstream g("date.out");
int x;
void cs(){
if(!f.eof()){f>>x;
             cs();
             g<<x<< ' ' ;}
}
void main()
{cs(); f.close();g.close();}
```

6. *Varianta Pascal*

```
function sub(nr:integer):integer;
var i:integer;
begin
for i:=2 to trunc(sqrt(nr)) do
  if nr mod i=0 then return 0;
return 1;
end;
 ...
 assign(f,'bac.txt');
 reset(f); read(f,n);
 for i:=1 to n do read(f,v[i]);
 close(f);  max:=0; k:=0;
 for i:=1 to n do
   if sub(v[i])=1 then k:=k+1
   else   begin
      if k>max then     begin
         max:=k; sf:=i-1; end;
      k:=0;
    end;
for i:=sf-max+1 to sf do
  write(v[i],' ,);
```

Varianta C/C++

```
int sub(int nr)
{  for (i=2;i<sqrt(nr);i++)
     if (nr%i==0) return 0;
   return 1;
}
....
ifstream f("bac.txt");
f>>n;
for (i=1;i<=n;i++) f>>v[i];
f.close(); max=k=0;
for (i=1;i<=n;i++)
  if (sub(v[i]))  k++;
  else
  {  if (k>max) { max=k;
                  sf=i-1;
                }
     k=0;
  }
for (i=sf-max+1;i<=sf;i++)
  cout<<v[i]<<" ";
```

5.1.

1. b	2. b	3. d	4. c	5. a	6. c
7. c	8. b	9. d	10. d	11. c	12. b
13. a	14. a	15. b	16. a	17. d	18. a
19. b	20. b	21. b	22. b	23. d	24. c
25. a	26. b	27. a	28. b	29. c	30. a
31. a	32. a	33. c	34. b	35. a	36. b
37. a	38. d	39. d	40. b	41. a	42. d
43. d	44. d	45. c	46. b	47. a	48. b
49. a	50. c	51. c	52. c	53. b	54. c
55. d	56. a	57. b	58. c	59. a	60. c

61. b	62. b	63. c	64. d	65. a	66. c
67. d	68. a	69. d	70. b	71. a	72. a
73. d	74. a	75. c	76. b	77. b	78. b
79. d	80. a	81. a	82. d	83. b	84. d
85. c	86. c	87. a	88. a	89. d	90. a
91. a	92. d	93. a	94. a	95. d	96. d
97. b					

5.2.

1. Se observă că numărul de numere care îndeplinesc cerinţele din enunţ este egal cu C_n^k. Va trebui implementată o funcţie care să calculeze această valoare într-un mod eficient din punct de vedere al timpului de executare. Pentru aceasta se poate scrie o funcţie care să implementeze formula $C_n^k = C_{n-1}^k + C_{n-1}^{k-1}$. Trebuie evitate apelurile repetate pentru aceleaşi valori ale parametrilor şi pentru aceasta se poate declara o matrice **a** care este iniţializată cu **0** şi în care, pe măsură ce se calculează, se vor reţine valori ale combinărilor (**a[n,k]**$\leftarrow C_n^k$). Utilizarea brută a formulei de calcul $C_n^k = \frac{n!}{k!(n-k)!}$ nu este recomandată datorită valorilor foarte mari pe care le poate lua **n!** Totuşi se pot da soluţii care pleacă de la această formulă, dar cu următoarele observaţii:

- se menţionează că programul va fi scris într-o versiune de limbaj care are tipuri întregi pe **64 de biţi** (de exemplu **gnuc, CBilder, Freepascal**)
- se simplifică expresia combinărilor cu factorul mai mare de la numitor; astfel, dacă **n-k>k** atunci rezultatul este **(n-k+1)(n-k+2)...n / k!**
  altfel rezultatul este **(k+1)(k+2)...n / (n-k)!**

2. Dacă înlocuim grupul de valori care trebuie să rămână alăturate cu o valoare generică, constatăm că trebuie să obţinem **12** permutări ale unei mulţimi formate din **6** elemente, fie acestea $a_1, a_2, ..., a_6$. Pentru a nu apela la backtracking, putem proceda astfel:

- o permutare este cea identică;
- **5** permutări obţinem prin **5** rotaţii succesive spre stânga (sau dreapta) cu câte o poziţie a celor 6 elemente
- o permutare obţinem interschimbând valorile a două dintre aceste elemente
  5 permutări obţinem prin **5** rotaţii succesive spre stânga (sau dreapta) cu câte o poziţie a celor şase elemente după efectuarea interschimbării.

6.1.

1. c	2. b	3. b	4. b	5. c	6. d
7. c	8. c	9. b	10. a	11. b	12. a
13. c	14. c	15. c	16. c	17. c	18. c
19. d	20. c	21. d	22. d	23. d	24. d
25. c	26. b	27. a	28. d	29. b	30. b
31. c	32. c	33. b	34. b	35. b	36. d
37. d	38. c	39. b	40. c	41. c	42. c
43. c	44. b	45. d	46. d	47. d	48. c
49. c	50. c	51. b	52. d	53. c	54. b
55. d	56. b	57. b	58. a	59. b	60. a
61. c	62. b	63. c	64. d	65. d	66. b
67. c	68. b	69. a	70. a	71. b	72. a
73. a	74.a				

6.2.

1. ***Varianta Pascal***

```
{cerinţa a)}

function sub(p:pnod; k:longint):word;
var a:word;
begin a:=0;
  while p<>NIL do begin
      if p^.nr<=k then inc(a);
      p:=p^.adr; end;
  sub:=a;
end;
{cerinţa b)}
var prim:pnod; a,b:longint;
.................
Begin readln(a,b);
 {.........creare lista}
  writeln(sub(prim,b)-sub(prim,a))
end.
```

Varianta C/C++

```
#include <iostream.h>
//cerinţa a)

int sub(nod *p, int k)
{ int a=0;
  for(;p!=NULL; p=p->adr)
    if(p->nr<=k)a++;
  return a;
}
 //cerinţa b)
void main()
{ nod *prim; long a,b;
  cin>>a>>b;
  ....... //creare lista
  cout<<sub(prim,b)-
sub(prim,a);
}
```

2. ***Varianta Pascal***

```
{cerinţa a)}
type pnod=^nod;
    nod=record
          info:char;
          adr:pnod
        end;
procedure creare(c:string;var p:pnod);
procedure listare(p:pnod);
{cerinţa b)}
var prim,q,r,p:pnod;
    c:string[30];
.....................
Begin
readln(c); creare(c,prim);
p:=prim;
q:=p^.adr;
   while q<>nil do
  begin
   if p^.info=q^.info
    then begin
       r:=q;
       q:=q^.adr;
       p^.adr:=q;
       dispose(r);
     end
     else begin
           p:=q;
           q:=q^.adr;
          end;
end; listare(prim) end.
```

Varianta C/C++

```
//cerinţa a)
struct nod
       { char info;
         nod *adr; };
void creare(char c[], nod *&p)
void listare(nod *p)
//cerinţa b)
#include <iostream.h>

void main()
{ nod *prim, *p, *q, *r;
  char c[31];
  cin.get(c,31);
  creare(c,prim);
  p=prim;
  q=p->adr;
  while(q)
  { if(p->info==q->info)
    { r=q;
      q=q->adr;
      p->adr=q;
      delete r;/free(p);   }
    else { p=q; q=q->adr;}
  }
  listare(prim);
}
```

3. ***Varianta Pascal***

```
{cerinţa a)}
type pnod=^nod;
     nod=record
           info:integer; adr:pnod
         end;
{cerinţa b)}
function min(p:pnod):integer;
 var q:pnod; m:integer;
 begin q:=p^.adr;m:=p^.info;
   while p<>q do begin
     if q^.info<m then m:=q^.info;
     q:=q^.adr; end;
   min:=m;  end;
```

Varianta C/C++

```
//cerinţa a)
struct nod
       { int info;
         nod *adr;   };
//cerinţa b)
int min(nod *p)
 { nod *q=p->adr;
   int m=p->info;
   while(p!=q)
    { if(q->info<m) m=q->info;
      q=q->adr;}
   return m;
 }
```

4. ***Varianta Pascal***

```
type pnod = ^nod;
     nod = record
           info:longint;
           urm:pnod end;
function elim2(p:pnod):longint;
var q:pnod;
begin elim2 := -1;
      if p=nil then exit;
      if p^.urm = nil then exit;
      q := p^.urm; elim2 := q^.info;
      p^.urm := q^.urm;
      dispose(q);
end;
```

Varianta C++

```
struct nod {long info;
            nod *urm;};

long elim2(nod *p)
{
 if (p==NULL) return -1;
 if (p->urm==NULL) return -1;
 nod *q = p->urm;
 long x = q->info;
 p->urm = q->urm;
 delete q;
 return x;
}
```

5. a) ***Varianta Pascal***

```
type adrelem = ^elem;
  elem = record num:integer;
                next : adrelem
         end;
procedure inserare(var
   prim:adrelem; nr:integer);
function numarare(prim:adrelem;
   nr:integer):integer;
```

Varianta C++

```
  struct elem
  {int num; elem *next;};

  void inserare(elem *&prim,
                int num);

  int numarare
        (elem *prim, int nr);
```

b) Parcurgem toate numerele întregi formate din maximum patru cifre şi pentru fiecare dintre ele determinăm apelând funcţia **numarare** de câte ori apare în lista iniţială; dacă apare de cel puţin două ori, îl adăugăm la a doua listă apelând subprogramul **inserare**:

Varianta Pascal

```
function gen2(p1:adrelem) :  adrelem;
var p2:adrelem; i:integer;
begin p2 := nil;
      for i := -9999 to 9999 do
       if numarare(p1,i)>=2 then
         inserare(p2,i)
      gen2  := p2;
end;
```

Varianta C++

```
elem *gen2(elem *p1)
{elem * p2 = NULL;
 int i;
 for(i=-9999;i<=9999;i++)
  if numarare(p1,i)>=2)
    inserare(p2,i);
 return p2;
}
```

6. *VARIANTA PASCAL*

```
type ref=^nod;
     nod=record
         info:integer; adr:ref;
     end;
procedure scrie(p:ref);
var q:ref;
begin
q:=p;
while q<>NIL do
 begin
  if ((q^.info div 100) mod 2=0)
   write(q->info,' ');
  q:=q^.adr;
 end;
end;
```

VARIANTA C/C++

```
struct nod
 {
  int info;
  nod *adr
 };
void scrie(nod *p)
{nod *q=p;
 while(q)
 {
  if ((q->info/100)%2==0)
   cout<<q->info<<" ";
  q=q->adr;
  }
}
```

7. *Varianta Pascal*

```
{cerinţa a)}
type reper=^nod;
     nod=record
       info:string[30];adr:reper;
         end;
procedure creare(var p:reper);
function
cautare(p:reper;x:string):reper;
var q:reper;
begin
 q:=p;
while (q^.info<>x) and (q<>nil) do
   q:=q^.adr;
cautare:=q;
end;
{cerinţa b)}
var p:reper;
begin
creare(p);
if cautare(p,"bacalaureat")<>nil
then
 write('cuvantul bacalaureat a fost
        gasit in lista')
 else
write('cuvantul bacalaureat NU a
       fost gasit in lista')
end.
```

Varianta C/C++

```
//cerinţa a)
typedef struct nod
   {char info[31]; nod* adr;
   }*PNOD;

void creare(PNOD &p);
 PNOD cautare(PNOD p, char x[31])
{
   PNOD q=p;
   while(strcmp(q->info,x)&& q)

     q=q->adr;
   return q;
}
//cerinţa b)
void main()
{PNOD p;
creare(p);
if(cautare(p,"bacalaureat"))
 cout<<"cuvantul bacalaureat a
        fost gasit in lista";
 else
cout<<"cuvantul bacalaureat  NU a
       fost gasit in lista";
} elimin(s1,s2,s);
```

8. O soluţie posibilă constă în parcurgerea în ordine alfabetică a majusculelor şi numărarea apariţiei fiecărei majuscule în fraza dată. Dacă numărul de apariţii este mai mare de 2 atunci se adaugă în listă (după ultimul nod al acesteia), un nou nod ce va memora informaţiile specificate în enunţ. Operaţia de numărarea a unui caracter într-o frază nu ne-am propus să o descriem deoarece este simplă şi poate fi abordată în moduri diferite ce presupun prelucrarea la nivel de caracter sau prelucrarea cu ajutorul subprogramelor specifice şirurilor de caractere.

```
citeşte fraza
┌ pentru c←'A','Z'execută
│ k←nr_aparitii(c,fraza)
│ ┌dacă k ≥2 atunci inserare_nod(prim,c,k)
│ └■
└■
afişare_lista(prim)
```

9. Pentru a determina numărul de vocale din cuvânt vom apela subprogramul **cre(cuv)** care va creea lista cu toate literele cuvântului, determinând numărul iniţial de caractere, apoi apelăm subprogramul sterg pentru fiecare dintre cele 5 vocale şi în final determinăm numărul de litere rămase apelând **lungime(p)**.

Varianta Pascal

```
type cuvant=string[50];
     adr=^nod;
     nod=record inf:char;
         next:adr end;
procedure cre (cuv:cuvant;var p:adr);
procedure sterg(var p:adr;c:char);
function lungime(p:adr): integer;
var p:adr;
   c,v:cuvant;n,i:integer;
begin readln(c); cre(c,p);
   n:=length(cuv); v:='aeiou';
   for i:=1 to 5 do
      sterg(p,v[i]);
   writeln(n-lungime(p));
end.
```

Varianta C/C++

```
struct nod {char inf;
            nod * next;};

void cre(char cuv[50], nod *&p);
void sterg(nod *&p,char c);
int lungime(nod *p);
void main()
{nod *p=NULL; int n;
 char c[50],v[]="aeiou";
 cin>>c; cre(c,p);
 n=strlen(c);
 for(int i=0;i<5;i++)
     sterg(p,v[i]);
 cout<<n-lungime(p);
 }
```

10. *Varianta Pascal*

```
type lista=^nod;
     nod=record
          info:integer; adr:lista
         end;
function s1(p:lista):integer;
var c:lista;  k:integer;
begin c:=p; k:=0;
repeat
  if c^.info * c^.adr^.info<0
          then k:=k+1;
  c:=c^.adr
until c^.adr = nil;
s1:=k
end;
```

Varianta C/C++

```
struct lista
         { int info;
           lista *adr;  };
int s1(lista *p)
{ lista *c;
 int k; c=p; k=0;
 do {
 if(c->info * c->adr->info<0)
           k=k+1;
   c=c->adr;
  } while (c->adr != NULL);
 return k; }
```

11. *Varianta Pascal*

```
type lista=^nod;
     nod=record
          info:integer; adr:lista
         end;
procedure s1(p:lista);
var q:lista;
begin q:=p;
```

Varianta C/C++

```
struct lista
         { int info
           lista *adr; };
void s1(lista *p)
{
  lista *q;
  q=p;
```

```
while(q^.adr<>nil)and
(q^.info<q^.adr^.info) do q:=q^.adr;
if(q^.adr=nil)
  then
   writeln('el. listei sunt ord.
                strict crescător')
   else  begin  q:=p;
   while(q^.adr^<>nil) and
 (q^.info>q^.adr^.info) do
           q:=q^.adr;
   if(q^.adr=nil) then
   writeln('el. listei sunt ord.
            strict descrescător')
   else
   writeln('el. listei nu sunt
             ordonate strict')
end;
end;
```

```
while (q->adr!= NULL &&
     q->info<q->adr->info)
          q=q->adr;
if(q->adr==NULL)
  cout<<"el. listei sunt
ordonate strict crescător";
else
{q=p;
 while (q->adr!= NULL &&
     q->info>q->adr->info)
          q=q->adr;
 if(q->adr==NULL)
  cout<<"el. listei sunt
ord. strict descrescător";
  else
   cout<<"el. listei nu
   sunt ordonate strict";
 }
}
```

12. Parcurgem lista cu 2 pointeri, **q** şi **r**. Pointerul **q** va parcurge lista element cu element, iar pointerul **r** va fi modificat concomitent cu pointerul **q**, insa va „sari" peste un element. Astfel, când **r** ajunge la sfârşitul listei sau la ultimul element, pointerul **q** va conţine adresa elementului din mijlocul listei.

Varianta Pascal

```
type adresa=^nod;
     nod=record
          info:integer;
          adr:adresa
         end;
function s1(p:adresa):lista;
var q,r:adresa;
begin
 q:=p;
 r:=p;
 while(r<>nil)and(r^.adr<>nil)do
  begin
   q:=q^.adr;
   r:=r^.adr^.adr
  end;
  s1:=q
end;
```

Varianta C/C++

```
 struct nod
         {int info;
          nod *adr; };
lista s1(lista *p)
 {  nod *q,*r;
  q=p; r=p;
  while (r!=NULL &&
      r->adr != NULL)
   {q=q->adr;r=r->adr->adr;}
   return q;
 }
```

13. *Varianta Pascal*

```
type adresa=^nod;
     nod=record
          info:real;
          adr:adresa
    end;
procedure s1(p:adresa);
var q,r,aux:adresa;
begin
 q:=p; r:=p^.adr;
```

Varianta C/C++

```
 struct nod
         { float info;
           nod *adr;
         };
void s1(nod *p)
{
 nod *q,*r,*aux;
 q=p;
 r=p->adr;
```

```
while(r<> nil) do
 begin
    if (q^.info*r^.info<0)
      then begin new(aux);
         aux^.info:=(q^.info +
                 r^.info)/2;
         aux^.adr:=r; q^.adr=aux;
      end;
    q:=r; r:=r^.adr
 end
end;
```

```
while (r!= NULL)
  {if(q->info*r->info<0)
    {aux=new nod;
     aux->info=
     (q->info +r->info)/2;
     aux->adr=r;
     q->adr=aux;
     }
  q=r; r=r->adr;
  }
}
```

14.

Varianta Pascal

```
type adresa=^nod;
     nod=record
          info:integer; urm:adresa
         end;
var prim,p:adresa;
procedure Insereaza (p:adresa);
 var aux:adresa;
begin new(aux);
 aux^.info:=trunc(sqrt(p^.info));
 aux^.urm:=p^.urm;
 p^.urm:=aux;
end;
begin
. . .
p:=prim;
while p<>nil do begin
 if sqrt(p^.info)=
      trunc(sqrt(p^.info))
  then begin
    Insereaza(p); p:=p^.urm; end;
  p:=p^.urm;
end;
. . .
end.
```

Varianta C/C++

```
 struct nod
         {int info;
          nod *urm;
         };
void Insereaza (nod *p)
{ nod *aux;  aux=new nod;
  aux->info=sqrt(p->info);
  aux->urm=p->urm;
  p->urm=aux;}
void main()
{ nod *prim,*p;
 . . .
 p=prim;
 while(p!=NULL)
 {if(sqrt(p->info) ==
     floor(sqrt(p->info)))
         {Insereaza(p);
          p=p->urm;
         }
  p=p->urm;
 }
. . .
}
```

15. Vom parcurge lista până la întâlnirea primului element ce memorează un număr prim sau până la sfârţitul listei. Subprogramul va returna valoarea pointerului cu care s-a parcurs astfel lista.

Varianta Pascal

```
type adresa=^nod;
     nod=record
          numar:integer; urm:adresa
         end;
function nrprim(n:integer):boolean;
function primulnrprim(p:adresa):adresa;
var q:adresa;
begin  q:=p;
  while(q<>nil) and
   not nrprim(q^.numar) doq:=q^.urm;
  primulnrprim:=q;
end;
```

Varianta C/C++

```
struct nod
          {int numar;
           nod *urm;};
int nrprim(int n);

nod *primulnrprim(nod*p)
{nod *q;
 q=p;
 while(q!=NULL &&
  !nrprim(q->numar))
     q=q->urm;
 return q;  }
```

16 Soluţia propusă apelează subprogramul **creare** pentru a forma o listă circulară cu toate literele unui cuvânt citit de la tastatură. Pentru a afişa toate permutările circulare ale cuvântului se apelează subprogramul **afişare** pentru fiecare nod al listei circulare.

Varianta Pascal

```
type adresa=^nod;
     nod=record
           inf:char; next:adresa
         end;
     cuvant=string[20];
function creare(cuv:cuvant): adresa;
procedure afisare(p:adresa);
var p,q:adresa; c:cuvant;
begin readln(c); p:=creare(c);q:=p;
if(q=nil)then
   writeln('lista este vida')
else
  repeat
   afisare(q); q:=q^.next
  until q=p;
end.
```

Varianta C/C++

```
struct nod{char inf;
           nod* next;};
nod *creare(char cuv[]);
void afisare (nod *p);

void main()
{ nod *p,*q;  char c[21];
  cin>>c; p=creare(c);
  q=p;
  if(p==NULL)
    cout<<"lista este  vida";
  else
     do { afisare(q);
          q=q->next
        } while (q!=p);
}
```

17. *Varianta Pascal*

```
type ref=^nod;
     nod=record
         info:integer; leg:ref;
         end;
procedure elimin(var p:ref);
var r,q:ref;
begin if p^.leg=p then
 begin dispose(p); p:=NIL; end;
else
 begin r:=p^.leg;
  while(r<>p) do begin
    while sqrt(r^.info)<>
          trunc(sqrt(r^.info)) do
     r:=r^.leg; {stergem r}
     q:=r^.leg; r^.info:=q^.info;
     r^.leg:=q^.leg; dispose(q);
    end; end; end;
```

Varianta C/C++

```
typedef struct nod
        { int info;
          nod *leg;}*PNOD;
void elimin(PNOD &p)
{ PNOD r,q;
 if (p->leg==p) //lista are un nod
  {delete p; p=NULL;}
 else
  { r=p->leg;
    while(r!=p)
   { while ( sqrt(r->info!=
             floor(sqrt(r->info)))
     r=r->leg;
     //stergem r
     q=r->leg; r->info=q->info;
     r->leg=q->leg; delete q;
   } } }
```

18. *Varianta Pascal*

```
type ref=^nod;
     nod=record
         info:integer;next:ref;
         end;
function verif(p:ref):byte;
var q:ref;
begin q:=p^.next;
 while(q^.next<>p)do q=q^.next;
 if p^.info=
    (q^.info+p^.next^.info->info)/2)
    then verif:=1
    else verif:=0;
end;
```

Varianta C/C++

```
typedef struct nod
        { int info;
          nod *next;
        }*PNOD;
int verif(PNOD p)
{  PNOD q=p->next;
 while(q->next!=p) q=q->next;
  if (p->info ==
   (q->info+p->next->info)/2.0)
return 1;
  else  return 0;
}
```

19. *Varianta Pascal*

```
type adresa=^nod;
     nod=record
          info:integer;
          prec,urm:adresa
     end;

procedure Sterge(var prim,
     ultim:adresa; p:adresa);
var q,p:adresa;
begin
 q:=p^.prec;
 r:=p^.urm;
 if q<>nil then  q^.urm:=r
           else  prim:=r;
 if r<>nil then r^.prec:=q
           else ultim:=q;
 dispose(p)
end;
```

Varianta C/C++

```
struct nod
         { int info;
           nod *prec,*urm;
         };
void Sterge(nod *&prim,
   nod *&ultim, nod *p,)
{nod *q,*r;
 q=p->prec;
 r=p->urm;
 if(q!=NULL)  q->urm=r;
 else         prim=r;
 if(r!=NULL)  r->prec=q;
 else         ultim=q;
 delete p;
}
```

20. a) *Varianta Pascal*

```
type adresa=^nod;
     nod=record
          c:char;
          urm,ant:pointer;
         end;

procedure creare(x:string;
var prim,ultim:adresa);
var p:adresa;i:integer;
begin
prim :=nil;
for i:=1 to length(x) do
 begin
   new(p);
   p^.c:=x[i];
   p^.urm:=nil;
   if (prim=nil) then
        begin
         p^.ant:=nil;
         prim:=p;
        end
   else
        begin
         p^.ant:=ultim;
         ultim^.urm:=p;
        end;
  ultim:=p;
end
end;
```

Varianta C/C++

```
typedef struct nod
        { char c;
          nod *urm,*ant;
        }*PNOD;
void creare
(char x[],PNOD &prim,PNOD &ultim)
{ PNOD p; int i;
  prim = NULL;
  for(i=0;i<strlen(x);i++)
   { p=new nod; p->c=x[i];
     p->urm=NULL;
     if(prim==NULL)
       { p->ant=NULL;
         prim=p;
       }
      else
     { p->ant=ultim;
       ultim->urm=p;
     }
    ultim=p;
   }
}
```

b) Apelăm subprogramul care crează lista, apoi parcurgem lista de la ultimul nod la primul, afişând pe ecran informaţia memorată în fiecare nod al listei.

Varianta Pascal

```
var prim,ultim:adresa;
    cuv:string[30];
procedure afisare(ultim:pointer);
var p:pointer;
begin p:=ultim;
while p<>nil do  begin
 write(p^.c,'  '); p:=p^.ant;end
end;
begin readln(cuv);
 creare(cuv,prim,ultim);
 afisare(ultim); end.
```

Varianta C/C++

```
void afisare(PNOD ultim)
{ PNOD p=ultim;
 while(p)
  {cout<<p->c<<" ";p=p->ant;}
}

void main()
{ cin>>cuv;
  creare(cuv,prim,ultim);
  afisare(ultim);
}
```

7.1.

1. d	2. b	3. b	4. b	5. c	6. a
7. c	8. c	9. d	10. c	11. d	12. c
13. b	14. a	15. a	16. a	17. c	18. d
19. d	20. d	21. a	22. d	23. b	24. b
25. d	26. a	27. d	28. d	29. d	30. a
31. d	32. c	33. c	34. b	35. d	36. a
37. d	38. b	39. d	40. a	41. d	42. c
43. c	44. c	45. b	46. d	47. a	48. a
49. c	50. d	51. d	52. d	53. b	54. c
55. d	56. b	57. a	58. d	59. d	60. d
61. d	62. d	63. a	64. a	65. c	66. a
67. b	68. d	69. b	70. b	71. b	

7.2.

1. b	2. d	3. a	4. b	5. d	6. d
7. a	8. d	9. b	10. a	11. b	12. d
13. a	14. a	15. b	16. b	17. b	18. d
19. c	20. a	21. d	22. c	23. d	24. d
25. d	26. d	27. c	28. d	29. b	30. a
31. b	32. a	33. a	34. b	35. a	36. c
37. d	38. d	39. c	40. b	41. d	42. b
43. b	44. c	45. d	46. b	47. a	48. d
49. c	50. d	51. a	52. d	53. d	54. c
55. b	56. b	57. a	58. c	59. a	60. d
61. d	62. a	63. a	64. a	65. a	66. a
67. a	68. b	69. c	70. b	71. b	

7.3.

1. d	2. d	3. d	4. a	5. d	6. d
7. d	8. b	9. c	10. b	11. a	12. a
13. b	14. b	15. d	16. a	17. a	18. a
19. d	20. b	21. b	22. c	23. c	24. a
25. a	26. d	27. d	28. a	29. b	30. c
31. a	32. b	33. d	34. d	35. c	36. b
37. c	38. a	39. c	40. b	41. d	42. d
43. a	44. d	45. b	46. a	47. c	48. b
49. c	50. d	51. c	52. a	53. a	54. c
55. d	56. a	57. c	58. d	59. a	60. d
61. a	62. b	63. a	64. b	65. d	

Cuprins

	Pag.	răspunsuri
1. Algoritmi elementari şi date din tipurile simple	3	
1.1. Teste grilă limbaj pseudocod	3	268
1.2. Teste grilă limbaj de programare (elemente de bază, Pascal | C/C++)	9	268
1.3. Probleme (programe pseudocod, Pascal, C, C++)	35	268
2. Tipuri de date structurate	73	
2.1. Tipul tablou unidimensional	73	
2.1.1. Teste grilă (limbaj pseudocod, Pascal | C/ C++)	73	293
2.1.2. Probleme (programe pseudocod, Pascal, C, C++)	81	293
2.2. Tipul tablou bidimensional	87	
2.2.1. Teste grilă limbaj (limbaj pseudocod, Pascal |C/C++)	87	297
2.2.2. Probleme (programe pseudocod, Pascal | C / C++)	92	297
2.3. Tipul şir de caractere	102	
2.3.1. Teste grilă (Pascal | C/ C++)	102	304
2.3.2. Probleme (Pascal | C/ C++)	106	305
2.4. Tipul înregistrare	110	
2.4.1. Teste grilă (Pascal | C/ C++)	110	309
2.4.2. Probleme (Pascal | C/ C++)	116	310
3. Fişiere	117	311
4. Subprograme	139	
4.1. Subprograme nerecursive	139	
4.1.1. Teste grilă (limbaj pseudocod, Pascal | C/ C++)	139	340
4.1.2. Probleme (programe pseudocod, Pascal | C / C++)	148	340
4.2. Subprograme recursive	166	
4.2.1. Teste grilă (Pascal | C/ C++)	166	360
4.2.2. Probleme (programe pseudocod, Pascal | C / C++)	186	360
5. Combinatorică şi tehnica Backtracking	188	
5.1. Teste grilă	188	362
5.2. Probleme	205	363
6. Structuri dinamice de date	206	
6.1. Teste grilă (Pascal | C/ C++)	206	363
6.2. Probleme (Pascal | C/ C++)	230	364
7. Grafuri	237	
7.1. Teste grilă - Grafuri neorientate	237	372
7.2. Teste grilă - Grafuri orientate	248	372
7.3. Teste grilă - Arbori	259	372
Răspunsuri	268	