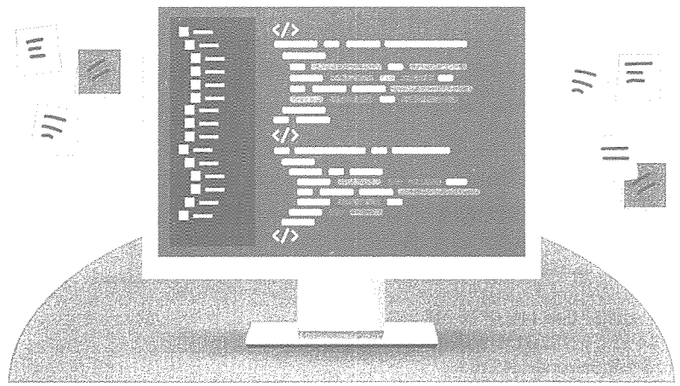
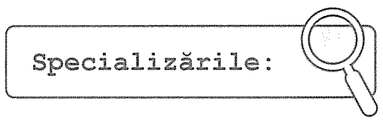
SILVIA GRECU
LUCIA MIRON
MIRELA ȚIBU

BACALAUREAT INFORMATICĂ
LIMBAJUL C++

Ghid complet de pregătire a examenului de Bacalaureat






```
{ Matematică-Informatică
  Științe ale Naturii
}
```

Editura Paralela 45

Lucrarea este realizată în conformitate cu programa examenului național de Bacalaureat
la disciplina Informatică.

Redactare: Iuliana Voicu
Tehnoredactare: Mariana Dumitru
Pregătire de tipar: Marius Badea
Design copertă: Mirona Pintilie

Editura Paralela 45

Descrierea CIP a Bibliotecii Naționale a României
GRECU, SILVIA
Bacalaureat - Informatică : limbajul C++ : ghid complet de pregătire a examenului de Bacalaureat : specializările Matematică-Informatică, Științe ale Naturii / Silvia Grecu, Lucia Miron, Mirela Țibu. - Pitești : Paralela 45, 2019
Conține bibliografie
ISBN 978-973-47-3115-2
I. Miron, Lucia
II. Țibu, Mirela-Anca
004

COMENZI – CARTEA PRIN POȘTĂ
EDITURA PARALELA 45
Bulevardul Republicii, nr. 148, Clădirea C1, etaj 4, Pitești,
jud. Argeș, cod 110177
tel.: 0248 633 130; 0753 040 444; 0721 247 918
tel./fax: 0248 214 533; 0248 631 439; 0248 631 492
e-mail: comenzi@edituraparalela45.ro

www.edituraparalela45.ro

Tiparul executat la tipografia Editurii Paralela 45
E-mail: tipografie@edituraparalela45.ro

Cuvânt-înainte

Lucrarea de față se dorește a fi un ghid de pregătire individuală a elevilor de liceu pentru proba de Informatică a examenului de Bacalaureat – varianta C++. În acest sens, cele unsprezece capitole de teorie prezintă un breviar al noțiunilor studiate la disciplina Informatică, prezente în programa de Bacalaureat, urmat de modele de teste propuse prin a căror rezolvare se poate realiza aprofundarea acestor noțiuni.

După parcurgerea noțiunilor de teorie, cartea se continuă cu teze întocmite după modelul examenului de Bacalaureat, atât pentru specializarea Matematică-Informatică, cât și pentru specializarea Științe ale Naturii.

Pentru a veni în sprijinul celor care vor folosi această carte, problemele și tezele au fost rezolvate integral, iar programele au fost verificate în Code::Blocks.

Ghidul poate fi utilizat în aceeași măsură și ca auxiliar didactic la orele de Informatică.

Sperăm că acest ghid se va dovedi un instrument util elevilor, dar și profesorilor care îl folosesc în pregătirea examenului de Bacalaureat.

Autoarele

PARTEA I

BREVIAR

CAPITOLUL 1

Reprezentarea algoritmilor în pseudocod. Elemente de bază ale limbajului C++

Teorie

Principiul programării structurate: Orice algoritm poate fi descris utilizând trei tipuri de structuri de control: liniară, alternativă și repetitivă.

Structuri de control – reprezentare în pseudocod	Implementare C++
• **Structura liniară** – conține operații de citire/scriere/atribuire. citește var_1, var_2, ..., var_n var_1, var_2, ... sunt identificatori de variabile scrie $expresie_1$, $expresie_2$, ..., $expresie_n$ $expresie_1$, $expresie_2$, var_2, ... sunt expresii de orice tip (numeric, caracter sau șir de caractere) *variabila* ← *expresie* Se evaluează expresia din partea dreaptă, iar valoarea obținută i se atribuie variabilei cu identificatorul *variabila.*	Citirea valorilor variabilelor de la tastatură: cin ≫ var_1 ≫ var_2 ≫ ⋯ ≫ var_n; //valorile variabilelor se citesc în ordinea din listă Afișarea valorilor expresiilor pe ecran: cout ≪ $expresie_1$ ≪ ⋯ ≪ $expresie_n$; //expresiile se vor evalua și afișa în ordinea din listă variabila = expresie; variabila < *op* > = expresie; unde *op* ∈ {+, −, ∗, /, %, ≫, ≪ , &, |, ^}, este echivalentă cu *variabila = variabila <op> expresie;*
Structura alternativă – conține operații de decizie: dacă <*expresie*> atunci *secvența_A* altfel *secvența_B*	Instrucțiunile *if* și *switch* implementează structura alternativă. if (*expresie*) instrucțiuni_A else instrucțiuni_B **Efect:** Se evaluează *expresie*. Dacă valoarea este diferită de 0, se execută secvența de

Sau

```
dacă <expresie> atunci
    secvența_A
```

Efect: Se evaluează *expresie*. Dacă este adevărată, se execută *secvența_A*, altfel se execută *secvența_B* și se continuă algoritmul cu următoarele structuri.

Secvențele A și/sau B pot conține orice alte structuri liniare, alternative sau repetitive.

Ramura altfel poate lipsi.

instrucțiuni_A, altfel se execută secvența de instrucțiuni_B.
În cazul în care o secvență conține mai mult de o instrucțiune, se va delimita prin { } grupul de instrucțiuni care definesc secvența.

Instrucțiunea *switch* pentru alternative multiple, simplifică modul de scriere pentru if-urile imbricate.

```
switch (expresie) {
    case constantă_1: instrucțiuni_1
                      break;
    case constantă_2: instrucțiuni_2
                      break;
    .....
    case constantă_n: instrucțiuni_n
                      break;
    default: instrucțiuni
    }
```

Efect: Valoarea *expresie* este evaluată la un tip întreg, apoi această valoare este comparată cu fiecare constantă. Este rulat blocul de instrucțiuni al valorii găsite.
Dacă este prezentă instrucțiunea *break* după blocul executat, se va părăsi instrucțiunea *switch*, altfel se vor executa în continuare toate blocurile de instrucțiuni ale constantelor care urmează, până la întâlnirea primului *break* sau până la sfârșitul instrucțiunii *switch*. În caz că numărul nu este egal cu niciuna dintre constante, este executat blocul aflat după *default*.

Structura repetitivă

- **Cu condiție inițială**

```
cât timp <expresie> execută
    secvența_A
```

Efect:
Pas 1. Se evaluează *expresie*.
Pas 2. Dacă este diferită de 0, se execută *secvența_A* și se reia Pas 1. Dacă este 0 se părăsește structura repetitivă și se continuă algoritmul cu următoarele structuri.

Instrucțiunea **while** implementează structura repetitivă cu condiție inițială.

```
while (expresie)
{
    //declarări de date locale
    Instrucțiuni
}
```

Variabilele declarate în blocul instrucțiunii *while* sunt vizibile doar în acest bloc și sunt valabile doar pe durata execuției instrucțiunii.

Obs. 1) Dacă la prima evaluare, valoarea *expresiei* este 0, *secvența_A* nu se execută deloc.

2) În *secventa_A* trebuie să se modifice valoarea cel puțin a unei variabile care apare în *expresie* astfel încât, după un număr finit de pași, execuția structurii *cât timp* să se încheie.

- **Cu condiție finală**

```
repetă
    secvența_A
până când <expresie>
```

Efect:
Pas 1. Se execută *secvența_A*.
Pas 2. Se evaluează *expresie*.
Pas 3. Dacă valoarea este diferită de 0 (este adevărată), se părăsește structura repetitivă și se continuă algoritmul cu următoarele structuri. Dacă valoarea este egală cu 0, se reia Pasul 1.
Obs. 1) *secvența_A* se execută cel puțin o dată, indiferent de valoarea inițială a *expresiei*.

2) În *secvența_A* trebuie să se modifice valoarea cel puțin a unei variabile care apare în *expresie* astfel încât, după un număr finit de pași, execuția structurii *cât timp* să se încheie.

- **Cu număr cunoscut de pași**

```
pentru contor ← start, stop, pas execută
    secvența_A
```

Obs. Dacă *pas* nu este precizat, implicit va avea valoarea 1.
Efect:
Pas 1. Se atribuie variabilei *contor* valoarea de start (evaluată ca număr întreg)
Pas 2. Se compară valoarea *contor* cu valoarea stop, în funcție de semnul pasului. Dacă pasul este 1 (parcurgere crescătoare) atunci compararea va fi *contor≤stop*, iar dacă pasul este –1 (parcurgere descrescătoare), atunci compararea va fi *contor≥stop*

Instrucțiunea **do-while** implementează structura repetitivă cu condiție finală.

```
do {
    //declarări de date locale
    Instrucțiuni
} while (not expresie);
```

Atenție! Condiția de oprire a buclei, exprimată de *expresie* din descrierea în pseudocod, se transformă în condiție de repetare a buclei, în instrucțiunea din C++. De aceea apare negația (not *expresie*).

Variabilele declarate în blocul instrucțiunii *do-while* sunt vizibile doar în acest bloc și sunt valabile doar pe durata execuției instrucțiunii.

Instrucțiuni pot fi orice instrucțiuni C++, inclusiv dintre cele care implementează structuri repetitive.

Instrucțiunea *for* implementează structura repetitivă cu număr cunoscut de pași *pentru*.

Dacă *pas* este 1

```
for (contor=start; contor<=stop;
contor++)
{
    Instrucțiuni
}
```

Pas 3. Dacă este adevărată evaluarea expresiei corespunzătoare *contor≤stop* (pentru pas = 1) sau *contor≥stop* (pentru pas = –1) se execută *secvența_A* și se trece la Pas 4.
Dacă este falsă, se părăsește structura repetitivă *pentru* și se continuă algoritmul cu următoarele structuri.
Pas 4. La valoarea *contor* se adună valoarea *pas* și se reia Pas 2.

Dacă *pas* este –1

```
for (contor = start; contor>=stop;
contor - -)
{
    Instrucțiuni
}
```

Sintaxa C++ a instrucțiunii **for** este

```
for (expresie1; expresie2; expresie3)
{
        Instrucțiuni
}
```

și este echivalentă cu

```
expresie1
while (expresie2)
{
        Instrucțiuni
        expresie3
}
```

Echivalența structurilor de control repetitive

Structura de control	Structuri de control echivalente
cât timp *<expresie>* execută *secvența_A*	dacă *<expresie>* atunci repetă *secvența_A* până când *<not expresie>* **Atenție!** Condiția de repetare exprimată prin *expresie* în structura *cât timp* devine condiție de repetare în structura *repetă-până când*, de aceea apare negația.
repetă *secvența_A* până când *<expresie>*	*secvența_A* cât timp *<not expresie>* execută *secvența_A* **Atenție!** Condiția de oprire exprimată prin *expresie* în structura *repetă-până când* devine condiție de repetare în structura *cât timp*, de aceea apare negația.

pentru contor ← start, stop execută *secvența_A*	contor←start cât timp *contor ≤ stop* execută *secvența_A* contor←contor+1
pentru contor ← start, stop execută *secvența_A*	contor←start dacă *contor ≤ stop* atunci repetă *secvența_A* contor←contor+1 până când *contor > stop*
pentru contor ← start, stop, –1 execută *secvența_A*	contor←start cât timp *contor ≥ stop* execută *secvența_A* contor←contor–1
pentru contor ← start, stop, –1 execută *secvența_A*	contor←start dacă *contor ≥ stop* atunci repetă *secvența_A* contor←contor–1 până când *contor < stop*
cât timp *val>0* execută *secvența_A* val←val–1	pentru val ← val, 1, –1 execută *secvența_A* sau pentru contor ← val, 1, –1 execută *secvența_A* **Atenție** la a doua implementare! În cazul în care *secvența_A* conține instrucțiuni care utilizează valoarea *val*, trebuie înlocuite toate referirile la val din *secvența_A* cu *contor*.

Elemente de bază ale limbajului C/C++

Datele prelucrate de un program sunt *constante* și *variabile*. Unei variabile i se poate modifica valoarea pe parcursul execuției blocului în care este declarată, în timp ce unei constante nu i se poate schimba valoarea.

O *variabilă* este caracterizată prin nume (identificator de variabilă), tip, spațiu de memorie alocat și valoare. Spațiul de memorie alocat (numărul de octeți) și domeniul valorilor posibile depind de tipul variabilei.
Identificatorul unei variabile poate fi format din litere 'a'...'z', 'A'...'Z', cifre '0'...'9' și simbolul '_' și nu poate începe cu o cifră.
Corect: _3ab, max2, M_aux, p123 **Incorect:** 1aB, 5_sum
În limbajul C/C++ pot fi reprezentate tipuri de date **simple** și **structurate** (omogene și neomogene).

Tipuri de date simple

Tip variabilă	Nr. octeți (bytes)	Domeniu de valori	Operatori	Funcții specifice
short int unsigned short int int = long int unsigned int = unsigned long int long long int unsigned long long	2 2 4 4 8 8	$[-2^{15}, 2^{15}-1]$ $[0, 2^{16}-1]$ $[-2^{31}, 2^{31}-1]$ $[0, 2^{32}-1]$ $[-2^{63}, 2^{63}-1]$ $[0, 2^{64}-1]$	Operatori: Aritmetici: +, −, *, /, % Pe biți: ≫, ≪, &, |, ^, ~ Relaționali: <,<=,>=, > De egalitate: ==, !=	<cmath.h> sqrt(x), x≥0, pentru $\sqrt{x}$ abs(x) pentru |x| pow(a,b) pentru a^b
float double	4 8	$[-3.2 \cdot 10^{38}, 3.2 \cdot 10^{38}]$ $[-1.7 \cdot 10^{308}, 1.7 \cdot 10^{308}]$	Operatori: Aritmetici: +, −, *, / Relaționali: <,<=,>=, > De egalitate: ==, !=	<cmath.h> sqrt(x), x≥0, pentru $\sqrt{x}$ abs(x) pentru |x| pow(a,b) pentru a^b
char unsigned char	1 1	−128, 127] [0, 255]	Operatori: Aritmetici: +, −, *, /, % Relaționali: <,<=,>=, > De egalitate: ==, !=	<ctype.h> islower(car) = { *1, car e literă mică*; *0, altfel* } isupper(car) = { *1, car e majusculă*; *0, altfel* } tolower(car) = *car* în minusculă = car +'a'−'A' toupper(car) = *car* în majusculă = car −'a'+'A'

Limbajul C/C++ permite realizarea de conversii implicite sau explicite de tip a varibilelor și constantelor.

Situații care determină efectuarea implicită a unei conversii într-o expresie:
- dacă operanzii sunt de același tip real sau întreg, dar se memorează în locații cu dimensiuni diferite, conversia se efectuează spre locația cu dimensiune maximă;
- dacă operanzii sunt de tip diferit, întreg și real, conversia se va aplica numerelor întregi care devin numere reale;
- dacă un operand de tip **char** intervine într-o expresie aritmetică se va realiza conversia către tipul întreg, iar valoarea cu care se va evalua expresia va fi codul ASCII al caracterului;
- dacă valoarea reală a unei variabile sau a unei expresii este atribuită unei variabile întregi, se va realiza o conversie implicită spre tipul întreg care va determina „trunchierea" părții zecimale a numărului real.

Exemplul 1

```
.....
int a=12, b, ok ;
long long int n;
char ch1='A', ch2;
ch2=ch1+3;
//codul ASCII('A')=65, 65+3=68 si ch2='D', codul ASCII('D')=68

b=a%7*1.5;
//se efectuează 12%7=5, apoi 5*1.5=7.5, care va fi trunchiat la partea întreagă 7, apoi atribuit lui b

ok=(a<b);
//se vor compara cele două valori ale variabilelor a și b (12<7) și se va atribui lui ok valoarea 0
cout <<ch2<<' '<<b<<' '<<ok;        //se vor afișa: D 7 0
```

Exemplul 2

```
.....
float x, y;
int a=7, b=2;
x=a/2+5.0;
        //se efectuează 7/2=3 (operanzi întregi) apoi 3+5.0=8.0 care se va atribui lui x
y=(float)a/2+3.0;     //se efectuează 7.0/2=3.5 apoi 3.5+3.0=6.5 care se va atribui lui y
cout <<x<<' '<<y;     //se vor afișa: 8  6.5
.....
```

Categorie	Operatori	Semnificație	Asociativitate
1. Prioritate unari de prioritate maximă	(), [], →, .	Apel de funcție Expresie cu indici Selectori de membru la structuri	St-Dr
2. Operatori unari	! ~, +, - ++, --, &, * sizeof(tip) (cast)	Negare logică Negare bit cu bit (complementare cu 1) Plus și minus unari Incrementare/decrementare (pre și post) Obținerea adresei/indirectare Dimensiune operand (în octeți) Conversie explicită de tip – cast	Dr-St
3. Operatori aritmetici multiplicativi	*, /, %	Înmulțire/împărțire/restul împărțirii întregi	St-Dr
4. Adunare, scădere	+, -	Plus și minus binari	St-Dr
5. Deplasări	<<, >>	Deplasare stânga/dreapta pe biți	St-Dr
6. Relaționali	<, <=, >, >=	Mai mic/ Mai mic sau egal/ Mai mare/ Mai mare sau egal	St-Dr
7. Egalitate	==, !=	Egal/ Diferit	St-Dr
8-10. Operatori logici pe biți	&, ^, |	ȘI/ SAU EXCLUSIV/ SAU logic bit cu bit	St-Dr
11-12. Operatori logici	&&, ||	ȘI/ SAU logic	St-Dr
13. Operator de atribuire compus	= <op>=	Atribuire simplă: variabila = expresie; Atribuire multiplă variabila1 = variabila2 = ... = expresie; Atribuire compusa: variabila < *op* >= expresie unde *op* ∈ {+, −, ∗,/, %, ≫, ≪, &, |, ^}, este echivalentă cu *variabila* = *variabila* < *op* > (*expresie*);	Dr-St
14. Operator condițional	?:	*expresieTest? variantaDa : variantaNu;* Se evaluează *expresieTest*. Dacă este adevărată (diferită de 0), se evaluează doar *expresieDa*, altfel, se evaluează doar *expresieNu*.	Dr-St
15. Virgula	,	*expresie1, expresie2,...expresieN*; Se evaluează expresiile în ordine iar valoarea finală este cea a *expresieN*	St-Dr

Expresiile sunt succesiuni de operatori și operanzi, corecte din punct de vedere sintactic. Orice expresie are o valoare, obținută la evaluarea acesteia.
În funcție de tipul operanzilor și al operatorilor, expresiile pot fi:
– aritmetice: valoarea obținută la evaluare este întreagă sau reală.
– logice: valoarea obținută la evaluare este 0 (fals) sau 1 (adevărat).
La evaluarea expresiilor logice se utilizează tabelele de valori ale operatorilor logici.

! (not), && (and), || (or)

!	
0	1
1	0

&&	0	1
0	0	0
1	0	1

||	0	1
1	0	1
0	1	1

În cazul expresiilor logice compuse se pot aplica regulile lui de Morgan:

!(E1 && E2) = !(E1) || !(E2)

!(E1 || E2) = !(E1) && !(E2)

unde E1 și E2 sunt expresii logice sau aritmetice.

Structura unui program C++

```
#include <iostream.h>
        //includerea altor librării necesare în program
using namespace std;
        //declarare variabile globale
        //declarare funcții utilizator
int main()
{
        //declarare variabile locale
        instrucțiuni
        return 0;
}
```

PROBLEME PROPUSE

1. Variabilele a și b reprezintă numere întregi nenule. Care dintre expresiile C/C++ au valoarea **1** dacă și numai dacă **a** și **b** au același semn și sunt impare?
   1) a+b>0 && a%2!=0 && b%2!=0
   2) !(a%2==0 || b%2==0) && !(a*b<0)
   3) a%2+b%2==2 && a*b>0
   4) (a%2>0 && b%2>0) || !(a%2+b%2<2)
2. Variabila **n** reprezintă un număr întreg nenul, iar variabilele **a** și **b** reprezintă numere reale. Care dintre următoarele expresii C/C++ au valoarea **1** dacă și numai dacă **n** este multiplu de **2019** și **nu** aparține intervalului **(a, b)**?
   1) n%2019==0 && n<a || n>b
   2) !(n%2019 || !(n<=a && n>=b))

3) floor(a)>=abs(n) && !(abs(n)<ceil(b) || n%2019!=0)
4) a>=n || b<=n && (n%3==0 && n%673==0)

3. Un număr natural este palindrom dacă este egal cu oglinditul său (De exemplu: **202**, **1881** sunt numere palindrom, iar **2012 nu** este palindrom). Știind că în variabila **x** este memorat un număr natural de exact 5 cifre, stabilește care dintre următoarele expresii C/C++ are valoarea **1** dacă și numai dacă numărul **x** este palindrom.
   a) x%100==x/100/10
   b) x/10%10+x%10*10==x/100/10
   c) x/1000%10==x/100%10 && x/10000==x%10
   d) x/10000==x/10%10 && x/1000==x%100

4. Știind că **x** este o variabilă care memorează un număr întreg nenul, precizează care este cea mai mică valoare ce se poate obține la evaluarea următoarei expresii C/C++: 200%x - x%5.
   a) -4 **b)** 0 **c)** -203 **d)** -195

5. Care este rezultatul ce se obține la evaluarea expresiei C/C++ de mai jos?
   (5*4%7-19/(7-4%5))/2
   a) 1 **b)** 0 **c)** 1.5 **d)** 1.1416

6. Se consideră următoarea secvență de instrucțiuni C-C++. Precizează câte operații de atribuire se vor efectua la execuția acesteia.

```
int n = 1023, k = 2;
while (n>0 || k>0)
{
   n = n / 10;
   k = k - 1;
}
```

   a) 10 **b)** 6 **c)** 8 **d)** 12

7. Se consideră **x** și **y** variabile care memorează numere naturale și următoarele 3 instrucțiuni de atribuire în C/C++. Precizează care este ordinea în care trebuie executate acestea, astfel încât să se realizeze interschimbarea valorilor celor două variabile.
   1) x = y - x **2)** y = x + y **3)** y = y - x
   a) 2, 3, 1 **b)** 1, 2, 3 **c)** 3, 1, 2 **d)** 3, 2, 1

8. Știind că **a**, **x** și **y** sunt variabile reale nenule, scrie instrucțiunea C/C++ corespunzătoare următoarei operații de atribuire: $a \leftarrow \frac{(x+y^2)\cdot(y-1)}{x^2\cdot y}$.

9. Știind că variabilele **x, y, z** memorează numere naturale, nenule și distincte două câte două, care dintre următoarele expresii C/C++ au valoarea **1** dacă și numai dacă numărul memorat în variabila **z** este fie multiplu de **x** și de **y**, fie este divizor al lui **x** și al lui **y**?
   a) !(z%x || z%y) && x%z==0 && z%y==0
   b) (z%x+z%y==0) || !(x%z || y%z)
   c) (x%z==0 || y%z==0) && y%x==0 && z%y==0
   d) z%(x*y)==0 || (x*y)%z==0

CAPITOLUL 2

Algoritmi elementari

Teorie

I. Algoritmi care prelucrează cifrele unui număr

- spargerea în cifre a numărului *n* și prelucrarea cifrelor de la dreapta la stânga:

```
dacă n=0 atunci
   prelucrare_cifra(0)

cât timp n≠0 execută
   cif ← n%10
   prelucrare_cifră(cif)
   n ← [n/10]
```

```
repetă
   cif ← n%10
   prelucrare_cifră (cif)
   n ← [n/10]
până când n=0
```

- spargerea în cifre a numărului *n* și prelucrarea cifrelor de la stânga la dreapta:

```
p ← 1
cât timp p*10≤n atunci
   p ← p*10

cât timp p≠0 execută
   cif ← [n/p]
   prelucrare_cifră(cif)
   n ← [n%p]
   p ← [p/10]
```

Formarea numărului cu cifrele din *n* de pe poziții impare, considerând numerotarea de la stânga la dreapta, prima cifră fiind considerată pe poziția 1.

```
p ← 1
cât timp p*10≤n atunci
   p ← p*10
nrNou ← 0
cât timp p≠0 execută
   cif ← [n/p]
   nrNou← nrNou*10+cif
   n ← [n%(p*10)]
   p ← [p/100]
```

prelucrare_cifră(cif) – descrie operațiile specifice problemei în care apare această prelucrare. De exemplu: suma cifrelor/numărarea cifrelor cu o anumită proprietate, verificarea proprietății de palindrom, eliminarea/inserarea cifrelor cu o anumită proprietate etc.

Atenție! Spargerea în cifre a unui număr distruge valoarea inițială a acestuia! Este necesară crearea unei copii a lui *n*, anterior spargerii, pentru a putea procesa ulterior valoarea inițială.

Exemple

*Construirea oglinditului și verificare **n** dacă este palindrom*	*Media aritmetică a cifrelor nenule*	*Cifra maximă din **n***
copie ← n oglindit ← 0 ┌cât timp *n ≠ 0* execută │ cif ← n%10 │ oglindit ← oglindit * 10 + cif │ n ← [n/10] └■ ┌dacă *copie=oglindit* atunci │ prelucrare_cifră(palindrom) └■	sum ← 0; nrcif ← 0; medie ← 0; ┌cât timp *n ≠ 0* execută │ cif ← n%10 │ ┌dacă *cif>0* atunci │ │ sum ← sum + cif │ │ nrcif ← nrcif + 1 │ └■ │ n ← [n/10] └■ ┌dacă *nrcif>0* atunci │ medie ← sum/nrcif └■ prelucrare(medie)	cifmax ← 0 ┌cât timp *n ≠ 0* execută │ cif ← n%10 │ ┌dacă *cif>cifmax* │ │ atunci │ │ cifmax ← cif │ └■ │ n ← [n/10] └■ prelucrare(cifmax)

*Eliminarea cifrelor impare din **n***	*Dublarea aparițiilor cifrelor pare din **n***	*Numărarea cifrelor pare existente în **n***
nrNou ← 0 p ← 1 ┌cât timp *n ≠ 0* execută │ cif ← n%10 │ ┌dacă *cif % 2 = 0* atunci │ │ nrNou ← nrNou + cif *p │ │ p ← p * 10 │ └■ │ n ← [n/10] └■ prelucrare(nrNou)	nrNou ← 0 p ← 1 ┌cât timp *n ≠ 0* execută │ cif ← n%10 │ ┌dacă *cif % 2 = 0* atunci │ │ nrNou ← nrNou + cif *p │ │ p ← p * 10 │ └■ │ nrNou ← nrNou + cif *p │ p ← p * 10 │ n ← [n/10] └■ prelucrare(nrNou)	nrPare ← 0 ┌repetă │ cif ← n%10 │ ┌dacă *cif % 2 = 0* │ │ atunci │ │ nrPare ← nrPare + 1 │ └■ │ n ← [n/10] └■ până când *n=0* prelucrare(nrPare)

Cifra de control a unui număr *n* se obține calculând suma cifrelor lui *n* apoi repetând procesul cu cifrele sumei obținute anterior până când se obține un număr format dintr-o singură cifră, numită cifră de control. De exemplu, pentru n=7912 se obțin pe rând sumele 7 + 9 + 1 + 2 = 19, 1 + 9 = 10, 1 + 0 = 1, iar 1 este cifra de control a lui 7912.

┌cât timp *n* > 9 execută
│ sumcif ← 0
│ ┌cât timp *n* ≠ *0* atunci
│ │ cif ← *n* % 10
│ │ sumcif ← sumcif + cif
│ │ *n* ← [*n*/10]
│ └■
│ n ← sumcif
└■
prelucrare(n)

Algoritmul eficient ca timp de execuție se bazează pe observația că cifra de control aunui număr este periodică și respectă relația:

$$\text{cifControl}(n) = \begin{cases} 0, \text{ dacă } n=0 \\ 9, \text{ dacă n\%9=0 și } n \neq 0 \\ n\%9, \text{ altfel} \end{cases}$$

II. Divizibilitate. Algoritmi care prelucrează divizorii proprii/improprii/primi ai unui număr

Fie *n* număr natural. Definim mulțimile:

Divizori **proprii**: d ∈ {2,3,4,...,[$\frac{n}{2}$]}, *n* : *d* Divizorii **primi**: d ∈ {2,3,5,7... [$\sqrt{n}$]}, *n* : *d*

Divizorii **improprii**: d ∈ {1,n}

*Divizorii proprii ai lui **n***	*Divizorii proprii ai lui **n** – optimizat*
p ← 1 ┌pentru d←2,[n/2],1 execută │ ┌dacă *n % d = 0* atunci │ │ prelucrare (d) │ └■ └■	p ← 1 ┌pentru d←2,[$\sqrt{n}$]– 1,1 execută │ ┌dacă *n % d = 0* atunci │ │ prelucrare (d) │ │ prelucrare (n/d) │ └■ └■ ┌dacă *d * d = n* atunci │ prelucrare (d) └■
Divizorii primi	*Divizorii primi ai lui **n** – optimizat*
d ← 2 ┌cât timp *n > 1* execută │ p ← 0 │ ┌cât timp *n% d = 0* execută │ │ n ← [n/d] │ │ p ← p + 1 │ └■ │ ┌dacă *p ≠0* atunci │ │ prelucrare (d, p) │ └■ │ d ← d+1 └■	d ← 2 ┌cât timp *d*d ≤ n* execută │ p ← 0 │ ┌cât timp *n% d = 0* execută │ │ n ← [n/d] │ │ p ← p + 1 │ └■ │ ┌dacă *p ≠ 0* atunci │ │ prelucrare (d, p) │ └■ │ d ← d+1 └■ ┌dacă *n ≠ 1* atunci │ prelucrare (n, 1) └■

Numere cu proprietăți speciale:

- N are număr impar de divizori ⇔ N este pătrat perfect (de exemplu, N = 36);
- N are exact 3 divizori ⇔ N este pătrat perfect de număr prim (de exemplu, N = 25);
- N este număr perfect ⇔ N este egal cu suma divizorilor mai mici decât N;
- A și B sunt numere prietene ⇔ A este egal cu suma divizorilor lui B, mai mici decât B, iar B este egal cu suma divizorilor lui A, mai mici decât A.

Formula lui Euler pentru calcularea numărului de divizori ai lui *n*, descompus în factori prim sub forma:

$n = p_1^{f1} p_2^{f2} \dots p_k^{fk}$, unde $p_1, p_2, \dots p_k$ sunt factorii primi la puterile $f_1\ f_2 \dots f_k$

$$\text{nrDivizori(n)} = (f_1 + 1)(f_2 + 1) \dots (f_k + 1)$$

III. Primalitate. Testarea primalității unui număr *n*

```
prim ← 1      //presupunem că n e prim
┌dacă n<2 atunci
│   prim ← 0
└■
┌pentru d←2,[√n],1 execută
│   ┌dacă n % d = 0 atunci
│   │      prim ← 0
│   └■
└■
┌dacă prim = 1 atunci
│      prelucrare (n)
└■
```

```
prim ← 1
d ← 2
┌cât timp prim=1 și d*d<=n execută
│   ┌dacă n% d = 0 atunci
│   │      prim ← 0
│   │   altfel
│   │      d ← d + 1
│   └■
└■
┌dacă prim = 1 atunci
│      prelucrare (n)
└■
```

IV. Cel mai mare divizor comun. Cel mai mic multiplu comun

```
┌cât timp b ≠ 0 execută
│   rest ← a%b
│   a ← b
│   b ← rest
└■
```

Cmmdc (a, b) – Algoritmul lui Euclid bazat pe împărțiri succesive (EFICIENT)

```
┌cât timp b ≠ a execută
│   ┌dacă a>b atunci
│   │      a ← a − b
│   │   altfel
│   │      b ← b − a
│   └■
└■
```

Cmmdc (a, b) – Algoritmul lui Euclid bazat pe scăderi succesive

Cmmmc (a, b) se poate determina folosind formula de calcul:

$$Cmmmc\ (a, b) = \frac{a * b}{cmmdc(a, b)}$$

Numerele a și b sunt **prime între ele** ⇔ *cmmdc (a,b)* = 1

V. Șiruri recurente

Generarea termenilor din șirul Fibonacci, definit de relațiile:

$$f_0 = f_1,\ f_n = \begin{cases} 1, \text{dacă } n = 1 \text{ sau } n = 0 \\ f_{n-1} + f_{n-2}, \text{dacă } n > 1 \end{cases}$$

```
F0 ← 1
F1 ← 1
Prelucrare(F0,F1)
┌pentru i ←2, n ,1 execută
│   F2 ← F1+F0
│   Prelucrare(F2)
│   F0 ← F1
│   F1 ← F2
└■
```

Prelucrarea primilor *n* termeni din șirul Fibonacci

```
F0 ← 1
F1 ← 1
┌cât timp n > 0 execută
│   F2 ← F1+F0
│   ┌dacă F2% 2 = 0 atunci
│   │      scrie F2
│   │      n ← n-1
│   └■
│   F0 ← F1
│   F1 ← F2
└■
```

Primii *n* termeni Fibonacci pari

VI. Baze de numerație. Conversii între baza 10 și baza b, 2≤b≤9, pentru un număr n

Conversie **din baza 10 în baza b**, realizată prin împărțiri succesive la b, cât timp n>0. Resturile, în ordine inversă obținerii lor, formează numărul în baza b, egal cu numărul n, în baza 10.

De exemplu: $75_{10} = 1001011_{(2)}$

Conversie **din baza b în baza 10**, realizată prin dezvoltare după puterile bazei b. Numărul n se sparge în cifre, care se înmulțesc cu puterile corespunzătoare ale bazei b.

De exemplu: $1001011_{(2)} = 1 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2 + 1 = 75_{(10)}$

```
repetă
    cif ← n%b
    Prelucrare(cif)
    n ← [n/b]
până când n=0
```

```
nrNou ← 0
p ← 1
cât timp n > 0 execută
    cif ← n % 10
    nrNou ← nrNou + p * cif
    n ← [n/10]
    p ← p * b
Prelucrare(nrNou)
```

Pentru baza b>10, mulțimea de simboluri care reprezintă alfabetul bazei este $\{0, 1, ..., 9, 'A', 'B', ...\}$. Resturile obținute în algoritmul de conversie din baza 10 în baza b, sunt cuprinse în intervalul $[0, b-1]$, iar cele mai mari decât 9 se vor înlocui cu litera corespunzătoare din alfabet.
Observație: Baza minimă în care un număr poate fi considerat corect reprezentat este cu 1 mai mare decât cifra minimă din număr.

Algoritmul de conversie din baza 10 în baza b, b>10

```
repetă
    rest ← n%b
    dacă rest < 10 atunci
        prelucrare ('0'+rest)
    altfel
        prelucrare ('A'+rest-10)
    n ← [n/b]
până când n=0
```

VII. Parcurgerea și prelucrarea numerelor dintr-un interval. Citirea și prelucrarea pe rând a *n* numere

Prelucrarea numerelor dintr-un interval $[a, b]$, a≤b sau $[1, n]$, parcurse crescător

```
citește a, b
pentru x←a, b, 1 execută
    prelucrare (x)
```

Prelucrarea a *n* numere citite pe rând

```
    citește n
pentru i ← 1, n, 1 execută
    citește x
    Prelucrare(x)
```

Prelucrarea perechilor dintr-un șir de *n* valori citite consecutiv


```
citește n
pentru x←1, n, 1 execută
    prelucrare (x)
```

Prelucrarea numerelor dintr-un interval $[a, b]$, a≤b sau $[1, n]$, parcurse descrescător

```
citește a, b
pentru x←b, a, -1 execută
    prelucrare (x)
```

```
citește n
pentru x←n, 1, -1 execută
    prelucrare (x)
```

```
    citește n, a
pentru i←2, n, 1 execută
    citește b
    Prelucrare(a,b)
    a←b
```

Exemple de optimizări în algoritmi de numărare a valorilor dintr-un interval, care au o anumită proprietate:

1) Numărul de multipli de k, mai mici sau egali decât **n**, este egal cu **n/k** ⟹ Numărul de multipli de k, din intervalul [a, b] este **b/k – (a – 1)/k**
2) Numărul de valori mai mici sau egale cu n, care au un număr impar de divizori, este $[\sqrt{n}]$, deoarece doar pătratele perfecte sunt valori cu număr impar de divizori ⟹ Numărul de valori din intervalul [a, b] care au un număr impar de divizori este $[\sqrt{b}] - [\sqrt{a-1}]$.
3) Numărul de valori x, $x \le n$ care au cifra de control k, $0 \le k \le 9$ este:

$$nr(n,k) = \begin{cases} 1, \text{ dacă } n = 0 \text{ și } k = 0 \\ 0, \text{ dacă } n = 0 \text{ sau } k = 0 \\ \left[\frac{n}{9}\right], \text{ dacă } n\%9 < k \\ \left[\frac{n}{9}\right] + 1, \text{ dacă } n\%9 \ge k \end{cases}$$

4) Numărul de valori x, x ∈ [a, b] care au cifra de control k, $0 \le k \le 9$ este nr(b, k) – nr(a–1, k).

PROBLEME PROPUSE

1. Se consideră programul pseudocod alăturat. S-a notat cu **x%y** restul împărțirii numerelor întregi **x** și **y** și cu **[z]** partea întreagă a numărului real **z**.
   a) Care este valoarea afișată pentru **a = 1372**?

```
citește a (număr natural)
i←0
a←a%10
```

b) Scrie cea mai mică valoare de 3 cifre care se poate citi pentru **a**, astfel încât valoarea afișată de algoritmul alăturat să fie maximă.
c) Scrie un algoritm echivalent cu algoritmul dat în care structura **cât timp... execută** să fie înlocuită cu o structură repetitivă cu test final.
d) Scrie programul C/C++ corespunzător algoritmului dat.

```
┌cât timp (a>1) și (a<10) execută
 i←i+1
 a←a*a
└■
scrie i*a
```

2. Se consideră programul pseudocod alăturat. S-a notat cu **[z]** partea întreagă a numărului real **z**.
a) Care este valoarea afișată la citirea următorului șir de valori **5 37 205 199 30 86**?
b) Scrie un șir de valori distincte, de cel mult două cifre pentru care algoritmul va afișa **9999**.
c) Scrie un algoritm echivalent cu algoritmul dat în care prima structura **cât timp... execută** să fie înlocuită cu o structură repetitivă **pentru**.
d) Scrie programul C/C++ corespunzător algoritmului dat.

```
citește n (număr natural nenul)
a ← 0
┌cât timp n>0 execută
 │  citește x (număr natural nenul)
 │  p ← 1
 │  ┌cât timp p≤[x/10] atunci
 │  │    p ← p * 10
 │  └■
 │   a ← a*10+[x/p]
 │   n ← n–1
└■
scrie a
```

3. Se consideră programul pseudocod alăturat. S-a notat cu **x%y** restul împărțirii numerelor întregi **x** și **y** și cu **[z]** partea întreagă a numărului real **z**.
a) Ce va afișa algoritmul alăturat dacă se citesc valorile **a=12 b=24**?
b) Scrie toate perechile de valori de câte o cifră care pot fi citite pentru a și b, astfel încât ultima pereche afișată de algoritmul alăturat să fie **6 7**.
c) Scrie un algoritm echivalent cu algoritmul dat în care prima structura **repetă... până când** să fie înlocuită cu o structură repetitivă cu condiție inițială.
d) Scrie programul C/C++ corespunzător algoritmului dat.

```
citește a, b (numere naturale nenule,
a<b)
p ← a; u ← b
┌cât timp p<u execută
 │  x ← p;   y ← u
 │  ┌repetă
 │  │    z ← x%y
 │  │    x ← y
 │  │    y ← z
 │  └■ până când y=0
 │  ┌dacă x=1 atunci
 │  │    scrie p, ' ', u, ';'
 │  └■
 │   p ← p+1;  u ← u–1
└■
```

4. Se consideră programul pseudocod alăturat. S-a notat cu **x%y** restul împărțirii numerelor întregi **x** și **y** și cu **[z]** partea întreagă a numărului real **z**.
a) Ce va afișa algoritmul alăturat dacă se citesc valorile **a=10011** și **b=2**?

```
citește a, b (numere naturale >0,
2≤b≤9)
m ← 0; x ← 0
```


b) Dacă pentru **b=8**, ce valoare poate fi citită pentru **a**, astfel încât algoritmul alăturat să afișeze **559**?
c) Scrie un algoritm echivalent cu algoritmul dat care să utilizeze o singură structură repetitivă.
d) Scrie programul C/C++ corespunzător algoritmului dat.

```
┌cât timp a > 0 execută
 │  p ← a % 10
 │  ┌pentru i ← 1, m execută
 │  │     p ← p*b
 │  └■
 │  x ← x+p
 │  a ← [a/10]
 │  m ← m+1
└■
scrie x
```

5. Se consideră programul pseudocod alăturat. S-a notat cu **x%y** restul împărțirii numerelor întregi **x** și **y** și cu **[z]** partea întreagă a numărului real **z**.
a) Ce va afișa algoritmul alăturat dacă se citesc valorile **5 2 32 110 4 243 512**?
b) Pentru **n=3, a=5** scrie un șir de valori distincte, de cel mult două cifre, ce pot fi citite pentru **b**, astfel încât algoritmul alăturat să afișeze **3**.
c) Scrie un algoritm echivalent cu algoritmul dat în care structura **cât timp... execută** să fie înlocuită cu o structură repetitivă cu condiție finală.
d) Scrie programul C/C++ corespunzător algoritmului dat.

```
citește n, a (număr natural nenul)
x ← 0
┌pentru i ← 1, n execută
 │  citește b (numere naturale
 │        0<a≤b)
 │  p ← 1
 │  ┌cât timp p<b execută
 │  │     p ← p*a
 │  └■
 │  x ← x+[b/p]
└■
scrie x
```

6. Se consideră programul pseudocod alăturat. S-a notat cu **x%y** restul împărțirii numerelor întregi **x** și **y** și cu **[z]** partea întreagă a numărului real **z**.
a) Ce va afișa algoritmul alăturat dacă se citesc valorile **x=2** și **y=10**?
b) Scrie o pereche de valori de două cifre ce pot fi citite pentru **x** și **y**, astfel încât algoritmul alăturat să afișeze la sfârșit valoarea **1**.
c) Scrie un algoritm echivalent cu algoritmul dat în care prima structură **cât timp... execută** să fie înlocuită cu o structură **pentru... execută**.
d) Scrie programul C/C++ corespunzător algoritmului dat.

```
citește x, y (numere naturale, 0<x≤y)
k ← 0
┌cât timp y≥x execută
 │  d ← 2
 │  ┌cât timp d*d≤y și y%d≠0 execută
 │  │     d ← d+1
 │  └■
 │  ┌dacă d*d=y atunci
 │  │     scrie y, ' '
 │  │     k ← k+1
 │  └■
 │   y ← y–1
└■
scrie k
```

7. Se consideră programul pseudocod alăturat. S-a notat cu **x%y** restul împărțirii numerelor întregi **x** și **y** și cu **[z]** partea întreagă a numărului real **z**.
   a) Ce va afișa algoritmul alăturat dacă se citește **n=120**?
   b) Scrie cea mai mică valoare de trei cifre ce poate fi citită pentru **n**, astfel încât algoritmul alăturat să afișeze **o singură valoare**, pentru fiecare dintre acestea.
   c) Scrie un algoritm echivalent cu algoritmul dat în care a doua structură **cât timp... execută** să fie înlocuită cu o structură repetitivă cu condiție finală.
   d) Scrie programul C/C++ corespunzător algoritmului dat.

```
citește n (număr natural nenule, 2≤n)
k ← 0
cât timp n>0 execută
    a←1; b←0
    cât timp b≤n execută
        b ← a+b
        a ← b–a
    scrie a, ' '
    n ← n–a
```

8. Se consideră programul pseudocod alăturat. S-a notat cu **x%y** restul împărțirii numerelor întregi **x** și **y** și cu **[z]** partea întreagă a numărului real **z**.
   a) Ce va afișa algoritmul alăturat dacă se citește **n=32153**?
   b) Câte numere de cel mult două cifre pot fi citite pentru **n**, astfel încât algoritmul alăturat să afișeze **1**?
   c) Scrie un algoritm echivalent cu algoritmul dat în care structura **cât timp... execută** să fie înlocuită cu o structură repetitivă cu condiție finală.
   d) Scrie programul C/C++ corespunzător algoritmului dat.

```
citește n (număr natural)
p←1;   x←0;
cât timp p < [y/p] execută
    x←x*10+[n/p] %10
    p←p*10
dacă x=[n/p] sau x=[n/(p/10)]
                sau n<10
  atunci  scrie 1
  altfel  scrie 0
```

9. Se consideră programul pseudocod alăturat. S-a notat cu **x%y** restul împărțirii numerelor întregi **x** și **y** și cu **[z]** partea întreagă a numărului real **z**.
   a) Ce va afișa algoritmul alăturat dacă se citește **n=4200**?
   b) Scrie două valori distincte, de cel puțin trei cifre, ce pot fi citite pentru **n** astfel încât algoritmul alăturat să scrie o valoare egală cu **n**, pentru fiecare dintre ele.
   c) Scrie un algoritm echivalent cu algoritmul dat în care a doua structură **cât timp... execută** să fie înlocuită cu o structură repetitivă cu condiție finală.
   d) Scrie programul C/C++ corespunzător algoritmului dat.

```
citește n (număr natural nenul, n>1)
  x ← 2; p ← 1
cât timp x≤n execută
    k←0;
    cât timp n%x=0 execută
        n←[n/x]
        k←k+1
    dacă k%2≠0 atunci
        p←p*x
    x←x+1
scrie p
```


Fișiere text

Teorie

Fișierul este o colecție de date omogene memorate pe un suport extern. Fișierul se identifică printr-un nume și o extensie. Într-un fișier text datele sunt memorate sub forma unei succesiuni de caractere (memorate prin utilizarea codului ASCII). Fișierul text este format din una sau mai multe linii, o linie se termină cu caracterul newline, mai puțin ultima care se termină cu marcajul de sfârșit de fișier (CTRL+Z).

Lucrul cu fișiere text

Pentru a lucra cu fișiere text trebuie să includem header-ul fstream.

```
#include<fstream>
```

Fiecărui fișier fizic îi corespunde în program un fișier logic (o variabilă), care trebuie declarată înaintea utilizării ei. Dacă un fișier text este utilizat numai pentru operații de citire acesta poate fi deschis astfel:

```
ifstream nume_logic(„nume_fizic");
```

Dacă un fișier text este utilizat numai pentru operații de scriere acesta poate fi deschis astfel:

```
ofstream nume_logic(„nume_fizic");
```

Citirea și afișarea datelor

Se consideră declarările următoare:

```
ifstream fin(„bac.in");
ofstream fout(„bac.out");
int x;
//citirea variabilei x din fisier
fin>>x;
//afișarea variabilei x în fișierul de ieșire
fout<<x;
```

Citirea datelor până la sfârșitul fișierului

Presupunem ca avem enunțul: Fișierul **bac.txt** conține valori întregi, acestea trebuie prelucrate, secvența <prelucrează x> realizează prelucrările dorite, aceasta va fi modificată conform cerințelor. Prelucrarea datelor dintr-un fișier când nu se cunoaște numărul de valori din fișier se poate face după un algoritm similar cu unul dintre algoritmii următori.

a)
```
ifstream fin(„bac.txt");
int x;
while(fin>>x)
  {<prelucrează x>}
```

```
b) ifstream fin(„bac.txt");
   int x;
   fin>>x;
   while(!fin.eof())
      {<prelucrează x>
       fin>>x;
      }
```

Testarea sfârșitului de fișier se poate face folosind funcția `eof()` care nu citește, ci doar testează dacă anterior a fost detectat sfârșitul de fișier. Apelul funcției se face astfel: `nume_logic.eof()`.

Închiderea fișierelor text

```
nume_logic.close();
```

Probleme propuse

1. Fișierul text `numere.in` conține pe prima linie un număr natural N, iar pe următoarele N linii câte o pereche de numere naturale a și b. Să se scrie un program care citește perechile de valori din fișier și afișează în fișierul `numere.out` pe linii diferite, pentru fiecare pereche din fișier cel mai mare divizor comun și cel mai mic multiplu comun al numerelor din pereche. Numerele din fișier au cel mult patru cifre.
2. Fișierul text `numere.in` conține pe prima linie cel mult 1 000 000 de numere naturale cu cel mult 9 cifre fiecare. Să se scrie un program care să determine și să afișeze pe ecran cele mai mici trei valori din fișier care sunt multipli ai numărului 3.
3. Fișierul text `numere.in` conține pe prima linie cel mult 1 000 000 de numere naturale cu cel mult 9 cifre fiecare, în ordine crescătoare. Să se scrie un program care să afișeze pe ecran pe aceeași linie, cu spațiu între ele, valorile distincte din fișier.
   Exemplu: Dacă fișierul `bac.txt` conține numerele **12 12 31 31 31 31 23** atunci pe ecran se afișează valorile **12 31 23.**
4. Numim **secvență10**, o secvență de numere aflate pe poziții consecutive în fișier cu proprietatea că toate numerele din secvență sunt numere divizibile cu 10. Fișierul text `numere.in` conține pe prima linie un număr natural N, iar pe următoarea linie un șir cu N numere naturale. Să se scrie un program care să se determine și să afișeze pe ecran lungimea maximă a secvențelor din fișier care sunt **secvență10**. Pentru determinarea valorii cerute se utilizează un algoritm eficient din punct de vedere al memoriei utilizate și al timpului de executare.
5. Fișierul text `bac.txt` conține pe prima linie cel mult 1 000 000 de numere naturale cu cel mult 9 cifre fiecare. Să se scrie un program care citește valorile din fișier și afișează pe ecran pe aceeași linie cu spațiu între ele valorile din fișier care sunt puteri ale lui 2.
6. Să se scrie un program C++ care citește de la tastatură două numere naturale a și b, ($a<b<10^9$) și afișează în fișierul `bac.out` toate numerele din intervalul `[a,b]` cu proprietatea că au toate cifrele egale.
   Exemplu: Dacă **a = 82 și b = 567** în fișier se vor afișa **88 99 111 222 333 444 555**.
7. Să se scrie un program C++ care citește de la tastatură două numere naturale a și b, ($a<b<10^9$) și afișează în fișierul `bac.out` toate numerele din intervalul [a,b] cu proprietatea că împărțite la suma cifrelor lor dau restul egal cu câtul.
   Exemplu: Dacă **a = 10 și b = 200** în fișier se vor afișa **16 39 55 96 187**.
8. Să se scrie un program C++ care citește de la tastatură un număr natural n și afișează în fișierul bac.out pe linii diferite toate prefixele numărului n.
   Exemplu: Dacă **a = 823451** în fișier se vor afișa pe linii diferite **823451 82345 8234 823 82 8.**
9. Fișierul text `numere.in` conține pe prima linie cel mult 1 000 000 de numere naturale cu cel mult 9 cifre fiecare. Să se scrie un program care să determine și să afișeze pe ecran separate printr-un spațiu ultima valoare din fișier care este multiplu al numărului 5 și numărul său de ordine.
   Exemplu: Dacă fișierul bac.txt conține numerele 12 13 **35** 123 **325** 61 **25** 54 **425** 511, atunci pe ecran se afișează valorile 425 **9.**

CAPITOLUL 4

Tablouri unidimensionale (vectori)

Teorie

Tabloul este un tip de date structurat în care elementele sunt de același tip, numit tip de bază al tabloului. Un tablou unidimensional se numește **vector**.

Declarare

```
tip nume[nr_max_elem];
```

Unde: `tip` este tipul elementelor tabloului; `nr_max_elem` este o expresie constantă ce reprezintă numărul maxim de elemente din tablou.

Exemplu:

```
int v[100], n, i;
```

Am declarat un vector cu 100 de elemente de tip întreg, putem utiliza doar *n* elemente din vector, n ≤ 100, spunem că *n* este numărul efectiv de elemente din vector.

Citirea și afișarea elementelor unui tablou

```
int v[100], n, i;
//citirea elementelor
cin>>n;
  for(i=0;i<n;i++)
  cin>>v[i];
  //afișarea elementelor pe aceeași linie/pe linii diferite
  for(i=0;i<n;i++)
  cout<<v[i]<<' ';//cout<<v[i]<<endl;
```

Algoritmi elementari cu vectori

```
int v[100], n, i, k, aux, x;
```

1. **Permutare circulară la stânga**

```
aux=v[0];
for(i=1;i<n;i++)
     v[i-1]=v[i];
v[n-1]=aux;
```


2. **Permutare circulară la dreapta**

```
aux=v[n-1];
for(i=n-2;i>=0;i--)
     v[i+1]=v[i];
v[0]=aux;
```

3. **Inserarea unui nou element în poziția k din vector**

```
for(i=n-1;i>=k;i--)
     v[i+1]=v[i];
v[k]=x;
n++;
```

4. **Ștergerea elementului din poziția k din vector**

```
for(i=k+1;i<n;i++)
     v[i-1]=v[i];
n--;
```

5. **Sortarea elementelor unui vector**

a) *Sortare prin interschimbare (metoda bulelor)*

Vectorul se parcurge în mod repetat comparând două elemente consecutive ale vectorului și interschimbându-le în cazul neîndeplinirii criteriului de ordonare (în cazul ordonării crescătoare dacă v[i] > v[i+1]). Astfel, elementele cu valoare mare sunt „împinse" către sfârșitul vectorului. Algoritmul se termină atunci când, la o parcurgere a vectorului nu s-a produs nicio interschimbare. Complexitatea algoritmului este $O(n^2)$. Considerăm că avem următoarea declarație de variabile:

```
int v[100], n, i, aux, sortat;
```

Secvența de program care ordonează crescător elementele vectorului prin metoda bulelor este:

```
do
{
     sortat = 1;
     for (i=0; i<n-1; i++)
          if (v[i] > v[i+1])
          {
               aux = v[i]; v[i] = v[i+1]; v[i+1] = aux;
               sortat = 0;
          }
} while (!sortat);
```

b) *Sortare prin selecția minimului (maximului)*

Sortarea prin selecție presupune determinarea elementului minim (maxim) din vector pentru fiecare poziție și plasarea acestuia pe poziția curentă, prin interschimbare. Complexitatea algoritmului este $O(n^2)$. Considerăm că avem următoarea declarație de variabile:

```
int v[100], n, i, aux, mini;
```

Secvența de program care ordonează crescător elementele vectorului prin metoda selecției minimului (analog se face și sortarea prin selecția maximului) este:

```
for (i=0; i<n-1; i++)
{
     mini = v[i];
     poz_min = i;
     for (j=i+1; j<n; j++)
          if (v[j] < mini)
          {
               mini = v[j];
               poz_min = j;
          }
     v[poz_min] = v[i];
     v[i] = mini;
}
```

c) *Sortare prin selecție directă*

```
for (i=0; i<n-1; i++)
     for (j=i+1; j<n; j++)
          if (v[i]>v[j])
          {
               aux = v[i]; v[i] = v[j]; v[j] = aux;
          }
```

6. Algoritmi de căutare

a) *Căutare liniară (secvențială)*

Operația de căutare constă în parcurgerea secvențială a vectorului și testarea egalității dintre elementul curent și valoarea căutată. Considerăm că avem următoarea declarație de variabile:

```
int v[100], n, i, x, poz;
```

Putem identifica două cazuri:

a.1. Elementele vectorului sunt într-o ordine oarecare, în această situație căutarea se oprește când am găsit elementul (căutare cu succes) sau când am epuizat elementele din vector (căutare fără succes).

```
poz = 0;
while ((x != v[poz]) && (poz < n))
          poz++;
if (poz < n)
     cout << x << " se afla pe pozitia " << poz;
else
     cout << x << " nu se gaseste in vectorul v";
```

a.2. Elementele vectorului sunt ordonate crescător (descrescător), în această situație căutarea se oprește când am găsit elementul (căutare cu succes) sau când am găsit primul element mai mare strict (mai mic strict) decât elementul căutat (căutare fără succes).

```
poz = 0;
while ((x < v[poz]) && (poz < n))
          poz++;
if (v[poz]==x)
     cout << x << "se afla pe pozitia " << poz;
else
     cout << x << "nu se gaseste in vectorul v";
```

b) *Căutare binară*

Vectorul în care se face căutarea trebuie fie ordonat. Ideea algoritmului: căutarea se efectuează printre valorile reținute de elementele de indice ls (limita stângă) și ld (limita dreaptă). Inițial ls = 0 și ld = n – 1. Procedeul constă în:

- se determină indicele elementului din mijloc mij = (ls + ld) / 2

Apar trei situații:

- dacă x = v[mij] atunci căutarea se termină cu succes;
- dacă x < v[mij] atunci valoarea va fi căutată între elementele de indice ls și mij – 1;
- dacă x > v[mij] atunci valoarea va fi căutată între componentele de indice m + 1 și ld.

Secvența de program care implementează algoritmul de căutare binară este:

```
ls=1;ld=n;ok=0;
while(ls<=ld&&!ok)
    {
        mij=(ls+ld)/2;
        if(v[mij]==d)ok=1;
        else if(d<v[mij])ld=mij-1;
        else ls=mij+1;
    }
if(ok)
    cout<<x<<" se afla in sir in pozitia "<<mij;
else
cout<<x<<" trebuie inserat pe pozitia "<<ls<<" astfel v
ramane ordonat".
```

7. Interclasarea a doi vectori ordonați

Interclasarea presupune obținerea din doi vectori ordonați (crescător sau descrescător) a unui singur vector ordonat. Considerăm că avem următoarea declarație de variabile:

```
int a[100], n, i, b[100], m, j, c[200], k;
```

Ideea algoritmului:

- se compară primul element din vectorul a cu primul din b și pe cel mai mic îl trecem în vectorul c, după care înaintăm în vectorul din are am preluat elementul;
- procesul se repetă până când se termină unul dintre vectori;
- în final se copiază la sfârșitul lui c toate elementele din vectorul rămas, acestea fiind ordonate.

Secvența de program care implementează algoritmul de interclasare este:

```
i = j = k = 0;
     while ((i < n) && (j < m))
     {
          if (a[i] < b[j])
               c[k++] = a[i++];
          else
               c[k++] = b[j++];
     }
     // daca au mai ramas elemente in a
     for (; i<n; i++)
          c[k++] = a[i];
     // daca au mai ramas elemente in b
     for (; j<m; j++)
          c[k++] = b[j];
```

8. Vector de frecvență/Vector caracteristic/vector de poziție

- Fiind dat un șir cu foarte multe elemente, valorile din șir fiind din mulțimea {0, 1, 2, …, vmax}, pentru a memora valorile din șir și a le prelucra în ordinea crescătoare (descrescătoare) a valorilor din șir putem utiliza un vector cu vmax+1 elemente, în care un element vf[i] reprezintă numărul de apariții (frecvența) a valorii i în șir, în acest caz particular avem un algoritm de ordonare de complexitate liniară O(n).
- Dacă ne interesează doar dacă o valoare a apărut sau nu în șir, putem utiliza un vector care memorează doar valorile 1 sau 0 după cum valoarea i aparține sau nu șirului.
- Sunt situații în care trebuie să memorăm poziția primei/ultimei apariții a valorii i în șir.

PROBLEME PROPUSE

1. În fișierul numere.in se află pe prima linie, N, un număr natural par, iar pe următoarele linii, se află N, numere naturale, reprezentând elementele unui vector, valorile din fișier au cel mult 4 cifre. Să se scrie un program care citește valorile din fișier și afișează pe ecran vectorul obținut prin permutarea circulară la stânga cu o poziție a elementelor din prima jumătate și prin permutareea circulară la dreapta cu o poziție a elementelor din a doua jumătate.
2. Să se scrie un program C++, care citește de la tastatură un număr natural N, cu cel mult 10 cifre, programul determină și afișează cel mai mare număr care se poate obține utilizând:
   a) toate cifrele lui N;
   b) toate cifrele distincte ale lui N.
3. În fișierul numere.in se găsesc pe prima linie cel mult 1 000 000 de numere naturale cu cel mult 2 cifre fiecare. Să se scrie un program care citește valorile din fișier și le afișează în fișierul numere.out în ordine crescătoare. Se va utiliza un algoritm eficient din punct de vedere al timpului de executare.
4. În fișierul numere.in se găsesc pe prima linie cel mult 1 000 000 de numere naturale cu cel mult 2 cifre fiecare. Să se scrie un program care citește valorile din fișier și afișează în fișierul numere.out în ordine crescătoare valorile distincte din fișierul numere.in. Se va utiliza un algoritm eficient din punct de vedere al timpului de executare.
5. În fișierul numere.in se găsesc pe prima linie cel mult 1 000 000 de numere naturale cu cel mult 2 cifre fiecare. Să se scrie un program care citește valorile din fișier și afișează în fișierul numere.out pe linii diferite toate valorile distincte din fișierul numere.in urmate de distanța dintre prima și ultima apariție, dacă o valoare apare o singură dată, distanța va fi 0, se va utiliza un algoritm eficient din punct de vedere al timpului de executare.
6. În fișierul numere.in se află pe prima linie, N (N < 500), un număr natural, iar pe următoarele linii, se află N, numere naturale, reprezentând elementele unui vector. Să se scrie un program care citește valorile din fișier și afișează pe ecran vectorul obținut prin inserarea valorii 2019 după fiecare element par.
7. În fișierul numere.in se află pe prima linie, N, un număr natural, iar pe următoarele linii, se află N, numere naturale, reprezentând elementele unui vector. Să se scrie un program care citește valorile din fișier și elimină din vector un număr minim de elemente astfel încât în vector să nu se afle două valori egale pe poziții consecutive. Vectorul astfel obținut se va afișa pe ecran.
8. Fișierul bac.txt conține pe prima linie numărul natural N (N < 500), pe a doua linie un șir de N numere întregi, cu cel mult 4 cifre fiecare, ordonat strict crescător, iar pe a treia linie un număr X de același tip cu elementele vectorului. Numerele de pe aceeași linie sunt separate prin câte un spațiu. Se cere să se afișeze pe ecran cel mai mare număr din șir mai mic sau egal cu X. Dacă nu există un astfel de număr, se afișează pe ecran mesajul nu exista. Pentru determinarea numărului cerut se utilizează un algoritm eficient din punct de vedere al timpului de executare.
9. Fișierul bac.txt conține pe prima linie două numere naturale, M și N (M, N < 500), iar pe fiecare dintre următoarele două linii câte un șir de M, respectiv N numere naturale ordonate crescător. Numerele aflate pe aceeași linie a fișierului sunt separate prin câte un spațiu. Se cere să se afișeze pe ecran, în ordine descrescătoare, toate numerele impare aflate în cele două șiruri. Numerele afișate sunt separate prin câte un spațiu, iar dacă nu există niciun astfel de număr, se afișează pe ecran mesajul nu există. Pentru determinarea numerelor cerute se va utiliza un algoritm eficient din punct de vedere al timpului de executare.

CAPITOLUL 5

Tablouri bidimensionale (matrice)

Teorie

Tabloul este un tip de date structurat în care elementele sunt de același tip, numit tip de bază al tabloului. Un tablou bidimensional se numește **matrice**.

Declarare

```
tip nume[nr_max_linii] [nr_max_coloane];
```

Unde: `tip` este tipul elementelor tabloului; `nr_max_linii` este o expresie constantă ce reprezintă numărul maxim de linii din matrice, `nr_max_coloane` este o expresie constantă ce reprezintă numărul maxim de coloane din matrice.

Exemplu:

```
int a[10][10], L, C, i, j;
```

Am declarat o matrice cu 10 linii și 10 coloane și elemente de tip întreg, putem utiliza doar L linii din matrice și C coloane din matrice, L≤10, C≤10. Spunem că L este numărul efectiv de linii din matrice și C este numărul efectiv de coloane din matrice.

Citirea și afișarea elementelor unei matrice

```
int a[10][10], L, C, i, j;
//citirea elementelor
cin>>L>>C;
  for(i=0;i<L;i++)
       for(j=0;j<C;j++)
            cin>>a[i][j];
  //afișarea elementelor matricei
  for(i=0;i<L;i++)
       {for(j=0;j<C;j++)
            cout<<a[i][j]<<' ';
   cout<<endl;
       }
```

Matrice pătratică

Se numește matrice pătratică, o matrice în care numărul de linii este egal cu numărul de coloane (L = C). Într-o matrice pătratică avem cele două diagonale, diagonala principală și diagonala secundară.

Diagonala principală

int A[6][6], L, i, j;

- Pentru L = 5 și numerotarea liniilor și coloanelor de la 1 la L, elementele de pe diagonala principală sunt evidențiate în matricea de mai jos.
- Pe diagonala principală se află elementele de forma **A[i][j], cu proprietatea că i = j**, sau elementele **A[i][i],** unde i = 1, L (pentru numerotarea liniilor și coloanelor de la 1 la L) sau i = 0, L – 1 (pentru numerotarea liniilor și coloanelor de la 0 la L – 1).
- **Sub diagonala principală** se află elementele A[i][j], cu `i>j`.
- **Deasupra diagonalei principale** se află elementele A[i][j], cu `i<j`.

A[1][1]				
	A[2][2]			
		A[3][3]		
			A[4][4]	
				A[5][5]

Diagonala secundară

int A[6][6], L, i, j;

Cazul 1: Numerotarea liniilor și coloanelor de la 1 la L

- Pentru L = 5 și numerotarea liniilor și coloanelor de la 1 la L, elementele de pe diagonala secundară sunt evidențiate în matricea de mai jos.
- Pe diagonala secundară se află elementele de forma **A[i][j], cu proprietatea că i + j = L + 1**, sau elementele **A[i][L + 1 – i],** unde i = 1, L.
- **Sub diagonala secundară** se află elementele A[i][j], cu `i+j>L+1`.
- **Deasupra diagonalei secundare** se află elemetele A[i][j], cu `i+j<L+1`.

				A[1][5]
			A[2][4]	
		A[3][3]		
	A[4][2]			
A[5][1]				]

- **Cazul 2: Numerotarea liniilor și coloanelor de la 0 la L – 1**
- Pentru L = 5 și numerotarea liniilor și coloanelor de la 0 la L – 1, elementele de pe diagonala secundară sunt evidențiate în matricea de mai jos.
- Pe diagonala secundară se află elementele de forma **A[i][j], cu proprietatea că i + j = L – 1**, sau elementele **A[i][L – 1 – i],** unde i = 0, L – 1.
- **Sub diagonala secundară** se află elementele A[i][j], cu `i+j>L-1`.
- **Deasupra diagonalei secundare** se află elemetele A[i][j], cu `i+j<L-1`.

				A[0][4]
			A[1][3]	
		A[2][2]		
	A[3][1]			
A[4][0]				]

Vecinii unui element

Într-o matrice, în funcție de poziția sa, un element poate avea 3, 5 sau 8 vecini. Pentru un element A[i][j], vecinii săi sunt reprezentați în matricea următoare.

A[i – 1][j – 1]	**A[i – 1][j]**	**A[i – 1][j + 1]**
A[i][j – 1]	**A[i][j]**	**A[i][j + 1]**
A[i + 1][j – 1]	**A[i + 1][j]**	**A[i + 1][j + 1]**

PROBLEME PROPUSE

1. Scrie un program C/C++ care citește de la tastatură un număr natural N (2<N≤20) și o valoare întreagă x, programul construiește în memorie un tablou bidimensional cu N linii și N coloane în care: toate elementele din prima linie și prima coloană au valoarea x; oricare alt element este obținut prin însumarea celor două elemente vecine cu el, aflate pe linia anterioară și pe aceeași coloană cu el, respectiv pe aceeași linie cu el și pe coloana anterioară, ca în exemplu. Programul afișează pe ecran tabloul obținut, fiecare linie a tabloului pe câte o linie a ecranului, elementele fiecărei linii fiind separate prin câte un spațiu.
   Exemplu: Pentru n = 4 și x = 1 pe ecran se afișează tabloul de mai jos.
   ```
   1 1 1 1
   1 2 3 4
   1 3 6 10
   1 4 10 20
   ```
2. Scrie un program C/C++ care citește de la tastatură două valori naturale nenule M și N (M≤10, N≤10) și apoi M*N numere naturale nenule cu cel mult 4 cifre fiecare, reprezentând elementele unei matrice cu M linii și N coloane. Programul determină apoi valorile maxime de pe fiecare linie a matricei și afișează pe ecran suma valorilor maxime determinate.
   Exemplu: Pentru M = 3, N = 4 și matricea
   32 **71** 3 5
   10 **12** 6 9
   74 55 6 3 se afișează pe ecran valoarea **157 (157 = 71 + 12 + 74).**
3. Scrie un program C/C++ care citește de la tastatură două valori naturale nenule M și N (M≤10, N≤10) și apoi M*N numere naturale nenule cu cel mult 4 cifre fiecare, reprezentând elementele unei matrice cu M linii și N coloane. Programul determină apoi valorile minime de pe fiecare coloană a matricei și afișează pe ecran suma valorilor minime determinate.

Exemplu: Pentru M = 3, N = 4 și matricea
32 71 **3** 5
10 12 6 9
74 55 6 **3** se afișează pe ecran valoarea **28 (28 = 10 + 12 + 3 + 3).**

4. Scrie un program C/C++ care citește de la tastatură un număr natural, N (N<10), apoi N numere naturale din intervalul [0, 10], reprezentând valorile elementelor aflate pe prima linie a unui tablou bidimensional cu N linii și N coloane. Programul construiește în memorie tabloul, inițializând celelalte elemente, astfel încât fiecare linie să se obțină prin permutarea circulară a elementelor liniei anterioare spre stânga, cu o poziție, ca în exemplu. Programul afișează pe ecran tabloul obținut, fiecare linie a tabloului pe câte o linie a ecranului, elementele de pe aceeași linie fiind separate prin câte un spațiu.
   Exemplu: Dacă se citesc numerele n = 4, `1 2 3 4` apoi se obține tabloul:
   ```
   1 2 3 4
   2 3 4 1
   3 4 1 2
   4 1 2 3
   ```
5. Scrie un program C/C++ care citește de la tastatură numere naturale din intervalul [2 100], în această ordine: N, apoi elementele unui tablou bidimensional cu N linii și N coloane, iar la final un număr x. Programul afișează pe ecran numărul valorilor aflate pe conturul tabloului (format din prima linie, ultima linie, prima coloană și ultima coloană) care sunt multipli ai numărului x.
   Exemplu: Pentru N = 4, tabloul următor și x = 4 se va afișa 4.
   ```
   5 2 4 3
   8 2 4 1
   6 4 1 5
   8 4 5 2
   ```
6. Scrie un program C/C++ care citește de la tastatură două numere naturale M și N (2≤M≤50, 2≤N≤50) și elementele unui tablou bidimensional cu M linii și N coloane, numere naturale cu cel mult patru cifre fiecare. Programul determină liniile care au toate elementele egale cu aceeași valoare și, pentru fiecare astfel de linie afișează pe ecran pe linii diferite numărul liniei separat printr-un spațiu de valoarea respectivă, iar dacă nu există astfel de valori, programul afișează pe ecran mesajul Nu există.
   Exemplu: Pentru M=3, N=4 și tabloul alăturat, se afișează pe ecran numerele `1 5`, iar pe linia următoare `3 8`.
   ```
   5 5 5 5
   8 2 4 1
   8 8 8 8
   ```
7. Într-un tablou bidimensional format numai din valorile 0 sau 1, spunem că două coloane sunt complementare, dacă oricare două elemente aflate pe aceeași linie sunt diferite. Scrie un program C/C++ care citește de la tastatură două numere naturale M și N (2≤M≤50, 2≤N≤50) și elementele unui tablou bidimensional cu M linii și N coloane, numere naturale

din mulțimea {0,1}, programul afișează pe ecran câte coloane complementare cu prima coloană sunt în matrice.
Exemplu: Pentru M = 3, N = 4 și tabloul alăturat, se afișează pe ecran 2.

```
0 1 1 1
1 1 0 0
1 1 0 0
```

8. Scrie un program C/C++ care citește de la tastatură un număr natural N (2<N≤15) și construiește în memorie un tablou bidimensional cu N linii și N coloane în care toate valorile naturale din intervalul [1,N*N] sunt așezate șerpuit ca în exemplu. Programul afișează pe ecran tabloul obținut, fiecare linie a tabloului pe câte o linie a ecranului, elementele fiecărei linii fiind separate prin câte un spațiu.
Exemplu: Pentru n = 4 pe ecran se afișează tabloul de mai jos.

```
1  2  3  4
8  7  6  5
9  10 11 12
16 15 14 13
```

9. Scrie un program C/C++ care citește de la tastatură un număr natural N (2<N≤15) și construiește în memorie un tablou bidimensional cu N linii și N coloane în care toate valorile naturale din intervalul [1,N*N] sunt așezate în spirală ca în exemplu. Programul afișează pe ecran tabloul obținut, fiecare linie a tabloului pe câte o linie a ecranului, elementele fiecărei linii fiind separate prin câte un spațiu.
Exemplu: Pentru n = 4 pe ecran se afișează tabloul de mai jos.

```
1  2  3  4
12 13 14 5
11 16 15 6
10 9  8  7
```


CAPITOLUL 6

Șiruri de caractere

Teorie

Pointer este o variabilă ce memorează adresa unei zone de memorie și se declară astfel:
tip_data *nume_pointer;
Un **șir de caractere** reprezintă o succesiune de caractere ASCII, care se încheie cu caracterul '\0'(NULL).

Declarare șir de caractere

a) char nume_sir[lungime_maximă]; lungime_maxima reprezintă numărul maxim de caractere din șir +1.
b) char *nume_sir;

Exemple
char s[101],*q;
La declarare, un șir de caractere poate fi și inițializat: char t[5]="abc".

a	b	c	NULL
t[0]	t[1]	t[2]	t[3]

Citirea unui șir de caractere
char s[lungime_maximă];
a) Dacă șirul nu conține spații se poate utiliza:
cin>>s;
b) Dacă șirul conține spații se utilizează una dintre variantele următoare:
1. cin.get(s,lungime_maxima,caracter_final);
caracter_final este implicit caracterul newline ='\n'.
Se citește șirul s până la întâlnirea caracterului final (fără să-l introducă în șir) sau până la citirea unui număr maxim de lungime_maxima-1 caractere și adaugă NULL la sfârșitul șirului.
2. cin.getline(s,lungime_maxima, caracter_final);
Se citește șirul s până la întâlnirea caracterului final sau până la citirea unui număr maxim de lungime_maxima-1 caractere, adaugă NULL la sfârșitul șirului și extrage caracter_final din fluxul de intrare (îl va omite la citirea următoare din program).

Afișarea unui șir de caractere
cout<<s;

Subșir obținut dintr-un șir de caractere
char s[10]="cd1*bc";

c	d	1	*	b	c	NULL
s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]

Exemple de subșiruri din șirul s: "d1", "1"etc. Pointerul **s + 2** indică subșirul care începe cu s[2] și se încheie cu NULL.: "1*bc"

Funcții predefinite pentru șiruri de caractere (definite în headerul **cstring**)
char s[lungime_maxima],s1[lungime_maxima1];
char s2[lungime_maxima2],separatori[lungime_maxima],*p,ch;

Operația realizată de funcție, apel funcție	Descrierea execuției funcției
Lungimea unui șir de caractere `strlen(s)`	Returnează lungimea șirului s, dat ca parametru.
Copierea unui șir de caractere `strcpy(s1,s2)`	Copiază șirul s2 în șirul s1 și returnează adresa s1.
Copierea unui subșir dintr-un șir de caractere `strncpy(s1,s2,nr)`	Copiază primele nr caractere din șirul s2 în șirul s1 și returnează adresa s1.
Compararea a două șiruri de caractere `strcmp(s1,s2)`	Compară s1 cu s2 și returnează valoarea 0, dacă șirurile sunt identice. <0, dacă s1<s2 >0, dacă s1>s2
Compararea a două șiruri de caractere, fără a face distincție între litere mari și litere mici `stricmp(s1,s2)`	
Concatenarea a două șiruri de caractere `strcat(s1,s2)`	Concatenează s2 la s1 și returnează adresa s1 (se reunesc șirurile s1 și s2 prin adăugarea s2 după ultimul caracter din s1, rezultatul reunirii se obține în s1).
Căutarea unui caracter într-un șir de caractere `strchr(s,ch)`	Dacă șirul s conține caracterul ch, funcția returnează adresa din s a primei apariții a caracterului ch. Dacă șirul s nu conține caracterul ch, funcția returnează NULL.
Căutarea unui subșir într-un șir de caractere `strstr(s1,s2)`	Dacă șirul s1 conține un subșir identic cu șirul s2, funcția returnează adresa din s1 a primei apariții a șirului s2. Dacă șirul s1 nu conține un subșir identic cu șirul s2, funcția returnează NULL.

Separarea unui șir de caractere în mai multe componente (entități) **a)** `strtok(s,separatori)` **b)** `strtok(NULL,separatori)`	Șirul s conține mai multe entități (cuvinte, numere, propoziții etc.), separate prin unul sau mai multe caractere, memorate în șirul separatori. Primul apel al funcției are forma **a**, funcția înlocuiește cu NULL primul separator ce apare în șirul s și returnează adresa primei entități din șir. La fiecare dintre următoarele apeluri ale funcției strtok se aplică forma **b** și se obține adresa următoarei entități din șir. Funcția strtok returnează NULL dacă nu se mai poate obține nicio entitate din șirul s.
Transformarea unui șir de caractere la litere mari `strupr(s)`	Toate literele mici din șirul s se transformă în litere mari, celelalte caractere din s nu se modifică.
Transformarea unui șir de caractere la litere mici `strlwr(s)`	Toate literele mari din șirul s se transformă în litere mici, celelalte caractere din s nu se modifică.

Funcții predefinite pentru verificarea apartenenței unui caracter la o categorie de caractere (definite în headerul **cctype**)
char c;

Operația realizată de funcție, apel funcție	Descrierea execuției funcției
Verificare literă mare `isupper(c)`	Returnează o valoare diferită de 0, dacă c este literă mare.
Verificare literă mică `islower(c)`	Returnează o valoare diferită de 0, dacă c este literă mică.
Verificare cifră zecimală `isdigit(c)`	Returnează o valoare diferită de 0, dacă c este cifră în baza 10.
Verificare literă sau cifră `isalnum(c)`	Returnează o valoare diferită de 0, dacă c este literă sau cifră.
Transformare în literă mare `toupper(c)`	Transformă caracterul c în literă mare și îl returnează.
Transformare în literă mică `tolower(c)`	Transformă caracterul c în literă mică și îl returnează.

PROBLEME PROPUSE

1. În fișierul `sir.in` pe prima linie se află un text **t** având cel mult 200 de caractere (litere și spații), format din mai multe cuvinte separate prin spațiu. Pe a doua linie din fișier se află un cuvânt **c**, format din cel mult 20 de litere. Scrie un program C++ care citește șirurile de caractere t și c și modifică în memorie șirul **s** prin eliminarea tuturor cuvintelor care au un prefix comun cu șirul **c**. Programul afișează pe prima linie din fișierul `sir.out` șirul s, obținut după modificare și pe a doua linie numărul de cuvinte eliminate.

   Exemplu

sir.in	sir.out
maculator calcul cub informatica calculator	maculator informatica 2

2. Se citește de la tastatură un text **t** , având cel mult 50 de caractere, format din litere și cifre egale cu 1 sau cu 2. Scrie un program C++ care citește șirul t și îl afișează pe ecran după modificarea sa în memorie, prin inserarea cuvântului **unu** după fiecare apariție a cifrei **1** în șirul t și inserarea cuvântului **doi** după fiecare apariție a cifrei **2** în șir. De exemplu, dacă t = "ab11cd21" atunci după execuția programului se obține t = "ab1unu1unucd2doi1unu".

3. Se citesc de la tastatură două șiruri de caractere **s** și **t**, având fiecare cel mult 250 de caractere, litere și cifre. Fiecare șir conține cel puțin o cifră și cel mult 4 cifre. Scrie un program C++ care determină și afișează pe ecran un număr, format din toate cifrele conținute de cele două șiruri, în ordinea apariției lor în șirul s și t. De exemplu, dacă s = "a1b2bd9" și t = "7dgh6j", se va afișa 12976.

4. În fișierul `cuvinte.in` pe prima linie se află un număr natural **n** (0<n<50). Pe fiecare dintre următoarele n linii se află un cuvânt având cel mult 30 de caractere, care sunt litere mici. Cuvintele din fișier sunt distincte. Scrie un program C++ care citește datele din fișierul de intarare și afișează în fișierul `cuvinte.out`, primul cuvânt și ultimul cuvânt în ordine alfabetică, dintre cuvintele citite. Fiecare linie din fișierul de ieșire va conține un cuvânt.

cuvinte.in	cuvinte.out
5 caiete calcule aparat cabinet calculator	aparat calcule

5. Se citește de la tastatură un text, având cel mult 200 de caractere, format din mai multe cuvinte, separate prin unul sau mai multe spații sau semne de punctuație: `.,?!`. Textul se încheie cu punct. Cuvintele conțin numai litere mici ale alfabetului englez. Scrie un program C++ care obține un nou text format din toate cuvintele textului citit care încep și se încheie cu o vocală, separate între ele printr-un spațiu. Programul va afișa pe ecran textul obținut sau va afișa mesajul Nu există, dacă textul citit nu conține astfel de cuvinte. De exemplu, dacă se citește textul „lumina, **asteroizi**, comete, **albine**, **oceane**, **oaza** soare, **alba.**" se va obține textul „asteroizi albine oceane oaza alba".

6. Se citește de la tastatură un șir de caractere **e**, având cel puțin 3 caractere și cel mult 200 de caractere care reprezintă o expresie aritmetică, formată din litere mici și cifre (care reprezintă **operanzi**), paranteze rotunde, paranteze pătrate și caractere speciale: `+-/*` (care reprezintă **operatori**). Expresia **e** este corectă dacă sunt îndeplinite următoarele condiții:
   – expresia nu conține doi operatori situați pe poziții consecutive;
   – un operator nu este precedat de o paranteză deschisă;
   – fiecare operand este format dintr-o literă mică sau o cifră;
   – expresia dată conține doar litere mici, cifre, paranteze și operatorii menționați anterior.
   În expresia dată, parantezele deschise și închise sunt grupate corect și există cel puțin un operator. Scrie un program C++ care citește șirul e și afișează pe ecran valoarea 1 dacă expresia este corectă și valoarea 0 în caz contrar.
   De exemplu, pentru `e=[(a+b)/2]*4+(b*c+5)` se va afișa 1, iar pentru `e=[(a+b)/*2]*4+(b*cd+5)` se va afișa 0.

7. Se citește de la tastatură o vocală **v**, un număr natural **n** (2≤n≤100) și un șir de caractere **s** format din **n** litere mici. Scrie un program C++, eficient ca timp de execuție, care afișează pe ecran secvența de lungime maximă din șirul s, care începe și se încheie cu vocala **v** sau afișează mesajul Nu există dacă șirul **s** nu conține o secvență cu aceste caracteristici. De exemplu, dacă v = 'a', n = 10, s = "**x**a**b**a**de**w**a**y**h**" se va afișa 7, iar dacă v = 'a', n = 5, s = "xabcd" se va afișa mesajul Nu exista.

8. Se citește de la tastatură un text **t**, având cel mult 400 de caractere, format din mai multe cuvinte, separate prin spațiu. Cuvintele din text sunt formate din litere. Scrie un program C++ care determină primul cuvânt din text care conține două vocale, situate pe poziții consecutive, afișează pe ecran acest cuvânt și numărul său de apariții în text sau afișează mesajul Nu există. De exemplu, dacă t = "Carte lectura **soare** copaci **soare** luna Soare soarele", se va afișa cuvântul soare 2.

9. În fișierul `cuvinte.in` pe prima linie se află două numere naturale **n** și **lg** (0<n<40, 1≤lg≤20). Pe fiecare dintre următoarele n linii se află un cuvânt având cel mult 30 de caractere, care sunt litere mici. Scrie un program C++ care citește datele din fișierul de intrare și afișează în fișierul `cuvinte.out`, în ordine alfabetică, cuvintele citite care au lungimea egală cu **lg.** Fiecare linie din fișierul de ieșire va conține un cuvânt.

cuvinte.in	cuvinte.out
5 6 caiete calcul aparat cabinet calculator	aparat caiete calcul

Structuri de date neomogene

Teorie

Tipul înregistrare este un tip de dată definit de utilizator, cu scopul de a reuni date de tipuri diferite care se referă la aceeași entitate (obiect, persoană, număr etc.). În limbajul C++ înregistrarea se definește printr-o structură.
O structură conține de regulă mai multe componente, denumite **câmpuri**.

Declararea unei structuri

struct nume_str{tip_data1 c_{11}[,c_{12}..];
tip_data2 c_{21}[,c_{22}..];
.....
tip_datak c_{k1}[,c_{k2}..];}[lista variabile];

c_{11},... c_{k2} reprezintă numele asociate câmpurilor.
Câmpurile pot avea un tip de dată predefinit, structurat (tablou).

Exemple

1. O structură care memorează coordonatele x și y ale unui punct din plan este:

```
struct punct
          { int x,y;
          } A,B,C;
```

A, B, C sunt variabile de tip **punct**, asociate pentru trei puncte din plan.

2.
```
struct complex
     { int re,im;
     } z;
```

definește un număr complex **z** în formă algebrică re+i*im, re este partea reală și im este partea imaginară a numărului z.

3.
```
struct student
     { char nume[20],facultate[40],grupa[6];
       int an_studiu,note[10];
     } st[100],*t;
```

Structura descrie datele unui student: nume, anul de studiu, grupa, facultatea și notele obținute la mai multe discipline. Variabila de tip tablou **st** memorează datele pentru cel mult 100 de studenți.

Structuri imbricate

O structură poate conține câmpuri ce aparțin unei alte structuri. De exemplu, pentru memorarea informațiilor pentru o persoană angajată se pot declara câmpuri de tip dată calendaristică.

```
  struct data
{ int zi,luna,an;
};
struct persoana
{char nume[20];
 data data_angajarii,data_absolvirii;
}p;
```

Operații cu structuri

- **Accesul unui câmp al structurii** se realizează prin notația:
  a) `Nume_variabila_struct.nume_câmp`, pentru o variabilă de tip struct
  b) `Pointer->nume_câmp` sau `(*pointer). nume_câmp`, pentru un pointer la struct.

Exemple, pentru structurile declarate anterior:

1. `A.x,A.y` (coordonatele punctului A)
2. `st[0].nume,st[0],grupa` (numele și grupa studentului cu indice 0 în vectorul st)
3. `st[0].note[0], ..., st[0].note[9]` (notele obținute de studentul cu indice 0 în vectorul st)
4. `t->nume,t->grupa` (numele și grupa studentului indicate prin pointerul t)
5. `p.data_angajarii.zi, p.data_angajarii.an` (ziua și anul angajării pentru persoana **p**), `p.data_absolvirii.an` (anul absolvirii pentru persoana **p**).

- **Operația de atribuire pentru structuri**
  `struct A{...} Variabila1, Variabila2;`
  `Variabila1=Variabila2`
  Se copiază toate câmpurile structurii definite prin Variabila2 în câmpurile corespunzătoare ca nume din Variabila1.

Probleme propuse

1. Se consideră declararea următoare:

```
struct melodie
{ char nume[30],autor[20], tara[20]; int loc_top,an_top;
}m[200];
```

Structura **melodie** memorează numele și autorul melodiei, țara în care s-a lansat melodia, cel mai bun loc ocupat în topuri din țara de lansare (loc_top) în anul precizat prin an_top. Vectorul **m** memorează datele pentru 200 de melodii.
Scrie o secvență de instrucțiuni C++ care conține declarații de variabile și instrucțiuni pentru:
a) determinarea și afișarea melodiilor din Top 3 în România în anul 2018;

b) afișarea melodiilor în ordine alfabetică după nume;
c) afișarea melodiilor în ordine alfabetică după țară și pentru aceeași țară în ordine crescătoare după loc_top.

2. Se consideră declararea următoare:

```
struct medalie
{char nume[20],judet[30], nume_medalie[10];
int an;}sp[500];
```

Structura **medalie** memorează numele și județul unui sportiv ce a obținut medalie la o competiție. Câmpul medalie reprezintă denumirea medaliei (are una dintre valorile aur, argint, bronz). Câmpul an indică anul competiției. Vectorul **sp** memorează datele pentru 500 de sportivi.
Scrie o secvență de instrucțiuni C++ care conține declarații de variabile și instrucțiuni pentru determinarea și afișarea numărului de medalii de aur, numărului de medalii de argint și a numărului de medalii de bronz obținute de sportivii din județul Iași în anul 2019.

3. Se consideră declarările următoare:

```
struct data
   {int zi,luna an;
   };
struct consultatie
  { char pacient[20],cabinet[40];
    data data_cons;
  }c[100];
```

Structura **data** memorează o dată calendaristică. Structura **consultatie** memorează numele unui pacient, cabinetul la care este programat pentru consultație și data programată a consultației (data_cons).
Vectorul **c** memorează datele pentru 100 de consultații. Scrie o secvență de instrucțiuni C++ care conține declarații de variabile și instrucțiuni pentru determinarea și afișarea numelor pacienților programați pentru consultații la cabinetul de stomatologie la data de 10 august 2019 și a numărului de pacienți programați la această dată.

4. Se consideră declarările următoare:

```
struct data
   {int zi,luna an;
   };
 struct excursie
    {char nume[30], oras[40],tara[30];
     data data_exc;
    }e[300];
char numed[30];
```

Structura **data** memorează o dată calendaristică. Structura **excursie** memorează numele persoanei înscrise pentru excursie, țara și orașul de destinație al excursiei și data de plecare în excursie (data_exc). Vectorul **e** memorează datele pentru 300 de persoane înscrise. Variabila **numed** reprezintă numele unei persoane.
Scrie o secvență de instrucțiuni C++ care conține declarații de variabile și instrucțiuni pentru determinarea și afișarea destinației (țară, oraș) și a datei pentru fiecare dintre excursiile la care s-a înscris persoana cu numele memorat prin variabila numed sau se va afișa mesajul Nicio excursie, dacă persoana nu s-a înscris la nicio excursie.

5. Se consideră declararea următoare:

```
struct angajat
{char nume[30], functie[20], departament[30];
int salariu_baza, ora_sp, nr_ore_sp, total_sporuri,salariu_obt;
 }a[300];
```

Structura **angajat** memorează numele, funcția îndeplinită, departamentul în care lucrează, salariul de încadrare (salariu_baza) al unui angajat. Alte câmpuri reprezintă: **salariu_obt** – salariul lunar obținut de angajat, **ora_sp** – suma plătită pentru o oră suplimentară, **nr_ore_sp** – număr de ore suplimentare efectuate de către angajat. Câmpul salariu_obt este necompletat inițial și se va completa cu suma dintre salariu_baza, total_sporuri și suma plătită pentru ore suplimentare. Vectorul **a** memorează datele pentru 300 de angajați ai unei firme.
Scrie o secvență de instrucțiuni C++ care conține declarații de variabile și instrucțiuni pentru:
a) completarea câmpului salariu_obt pentru toți angajații;
b) afișarea salariului de valoare maximă obținut de către angajații din departamentul producție;
c) afișarea numărului total de ore suplimentare efectuat de către angajații din departamentul marketing.

6. Se consideră declararea următoare:

```
struct student
{ char nume[20],grupa[10],bursa[3];
  int an_studiu, nr_credite;
}st[300];
```

Structura **student** memorează numele, grupa, anul de studiu al unui student, numărul de credite obținute la disciplinele de studiu (nr_credite). Câmpul bursa este necompletat inițial și se va completa cu valoarea "Da", dacă studentul va primi bursă sau "Nu", în caz contrar. Condiția de primire a bursei este ca studentul să aibă minimum 30 de credite obținute. Vectorul **st** memorează datele pentru 300 de studenți.
Scrie o secvență de instrucțiuni C++ care conține declarații de variabile și instrucțiuni pentru:
a) completarea câmpului bursa pentru studenți, conform condițiilor anterioare;
b) afișarea datelor studenților (nume, an studiu, grupa) pentru studenții care vor primi bursă.

7. Se consideră declarările următoare:

```
struct examinare
{char tip_examinare[10],disciplina[20];
  int nr_credite;
};
```

```
struct student
{ char nume[20],grupa[10];examinare e[10];
  int an_studiu, nr_total_credite;
}st[200];
```

Structura **examinare** memorează disciplina de studiu, tipul de examinare al disciplinei (scris, oral, proiect) și număr de credite obținute la examinare ∈ [0,7]. Structura **student** memorează numele, grupa, anul de studiu, numărul total de credite obținute din toate examinările denumit **nr_total_credite**, care nu este completat inițial și numărul de credite obținut la fiecare disciplină (dat de vectorul e). Vectorul st memorează informații pentru 200 de studenți.
Scrie o secvență de instrucțiuni C++ care conține declarații de variabile și instrucțiuni pentru:

a) completarea câmpului nr_total_credite pentru toți studenții;
b) afișarea datelor studenților (nume, an studiu, grupa, disciplina) care au obținut credite la proiecte;
c) afișarea numărului total de credite de valoare maximă obținut de studenți.

8. Se consideră declarările următoare:

```
struct data
   {int zi,luna an;
   };
struct spectacol
 { char denumire[30],autor[20];data data_sp;
   int nr_bilete, pret;
 }s[50];
```

Structura **data** memorează o dată calendaristică. Structura **spectacol** memorează denumirea, autorul, data desfășurării unui spectacol de teatru (data_sp). Câmpul nr_bilete reprezintă numărul de bilete vândute pentru spectacol. Câmpul preț reprezintă prețul unui bilet la spectacol. Vectorul sp memorează date despre 50 de spectacole desfășurate.
Scrie o secvență de instrucțiuni C++ care conține declarații de variabile și instrucțiuni pentru:

a) determinarea și afișarea sumei încasate din vânzarea biletelor pentru toate spectacolele desfășurate în luna mai din anul 2019;
b) determinarea și afișarea datelor calendaristice în care au fost susținute spectacole cu piese de I.L.Caragiale.

9. Se consideră declarările următoare:

```
struct data
   {int zi,luna an;
   };
 struct conferinta
   { char denumire[30],oras[20],tara[25],tematica[20];
     data data_conf; char nume[20],tip_inreg[20];
   } c[100];
```

Structura **data** memorează o dată calendaristică. Structura **conferința** memorează prin vectorul **c**, date despre participanții la o conferință:
– denumirea, tematica (ex.: IT, medicină, matematică etc.), data (data_conf) și locul de desfășurare (oraș și țară) al conferinței;
– nume reprezintă numele unui participant la conferință;
– tip_inreg reprezintă tipul de înregistrare la conferință: lector sau participant.
Scrie o secvență de instrucțiuni C++ care afișează:

a) numele lectorilor care au prezentat articole în anul 2019, la conferințe IT susținute în România, SUA și Japonia;
b) numele participanților din luna aprilie, anul 2019, la conferințe de medicină susținute la Iași și numărul lor.

CAPITOLUL 8

Subprograme (Funcții)

Teorie

Subprograme definite de utilizator

Orice subprogram (modul) este format din **antet** și **corpul subprogramului.**

Definiția completă a unui subprogram (modul apelat)

```
Tip_subprogram nume_subprogram (listă de parametri formali)
Corp subprogram
```

Tipul subprogramului poate fi un tip de date predefinit, un pointer sau void (funcție fără tip). Corpul subprogramului conține declarații de variabile, constante, tipuri de date și instrucțiuni. Lista de parametri poate fi vidă.

Declarația unui subprogram (prototip al subprogramului) conține antetul său:

```
Tip_subprogram nume_subprogram (listă de parametri)
```

Subprogram (modul) apelant reprezintă subprogramul în care se va utiliza subprogramul definit. Subprogramele apelant și apelat pot comunica prin parametri.

Apelul unui subprogram de tip void se realizează prin instrucțiune procedurală:

```
nume_subprogram (listă de parametri actuali (efectivi))
```

Apelul unui subprogram de tip predefinit sau pointer este un operand utilizat într-o instrucțiune expresie din subprogramul apelant de forma:

```
nume_subprogram (listă de parametri actuali (efectivi))
```

Parametrii formali ai unui subprogram apelat

1. **parametrii de intrare** prin intermediul cărora subprogramul apelat poate primi date din subprogramul apelant, un parametru se declară: **tip_data nume_parametru**;
2. **parametrii de ieșire** prin intermediul cărora subprogramul apelat transmite date subprogramului apelant, un parametru se declară: **tip_data &nume_parametru**;
3. **parametrii de intrare-ieșire** prin intermediul cărora subprogramul apelat poate primi date din subprogramul apelant pe care le modifică și le transmite subprogramului apelant, un parametru se declară: **tip_data &nume_parametru**.

Parametrii efectivi ai unui subprogram apelant

La apelul subprogramului, parametrii formali se pun în corespondență cu parametrii efectivi. Pentru un parametru formal de intrare, parametrul efectiv corespondent va fi o expresie compatibilă ca tip de dată. Pentru un parametru formal de ieșire sau de intrare-ieșire parametrul efectiv corespondent va fi o variabilă compatibilă ca tip de dată.

Parametrii formali și parametrii efectivi trebuie să corespundă ca număr, ordine în lista de parametri și tipuri de date.

Execuția unui subprogram

Apelul unui subprogram are următoarele etape:

– se întrerupe execuția modulului apelant, se salvează adresa de revenire din acest modul;

– se execută subprogramul apelat căruia i se alocă o zonă de memorie în stivă, utilizată pentru memorarea valorilor parametrilor de intrare (**transfer prin valoare**), adreselor transmise prin parametri de ieșire sau de intrare-ieșire (**transfer prin referință**) și variabilelor proprii (**variabile locale**). La încheierea execuției subprogramului, zona din stivă se eliberează;

– se reia execuția modulului apelant și se continuă cu instrucțiunea de la adresa de revenire.

Instrucțiunea return asigură comunicarea valorii unui subprogram apelat către subprogramul apelant și încheierea execuției subprogramului apelat. Are forma:

```
return expresie
```

Subprograme de tip int

- **Subprogram pentru determinarea sumei cifrelor unui număr natural n**

```
int suma (int n)//antet
//corpul subprogramului
   { int s=0;
    while(n)
    {s=s+n%10;
     n=n/10;
    }
    return s;
   }
```

- **Subprogram pentru verificarea primalității unui număr natural n**

```
int prim(int n)
   {int d;
    if(n<2 ||(n>2 && n%2==0))
     return 0;
    if(n==2)
     return 1;
    for(d=3;d*d<=n;d=d+2)
    {if(n%d==0)
      return 0;
    }
    return 1;
   }
```

Subprograme de tip void

- **Subprogram care furnizează prin parametrul cmd cel mai mare divizor comun a două numere naturale nenule**

```
void cmmdc(int a,int b,int &cmd)
{int r;
do
 {r=a%b;
   a=b; b=r;
 }
while(r!=0);
cmd=a;
}
```

- **Subprogram care furnizează prin parametrul vmax cel mai mare element al unui vector v și furnizează prin parametrul nr numărul de apariții a valorii maximului în vector**

```
void maxim(int n,int
v[],int &vmax,int &nr)
{int i; vmax=v[0];nr=1;
for(i=1;i<n;i++)
if(v[i]>vmax)
{vmax=v[i];nr=1;}
else
if(v[i]==vmax)
  nr++;
}
```

Apelul funcției în funcția main este:

```
int main()
{
int vmax,nr,n,v[nmax];
.....
maxim(n,v,vmax,nr);}
```

De exemplu,pentru n=4, v=(1,4,3,4), la apelul funcției se obține:

zona alocată în stivă funcției maxim		zona alocată în stivă funcției main
Adresa nr	→	Valoare nedefinită nr
Adresa vmax	→	Valoare nedefinită vmax
Adresa tablou v	→	v = (1,4,3,4)
n: valoarea 4		n: valoarea 4

La încheierea execuției funcției maxim se obține:

zona alocată în stivă funcției maxim		zona alocată în stivă funcției main
Adresa nr	→	Valoare nr = 2
Adresa vmax	→	Valoare vmax = 4
Adresa tablou v	→	v = (1,4,3,4)
n: valoarea 4		n: valoarea 4

Subprograme (funcții) predefinite (definite în headerul cmath)

Operația realizată de funcție, apel funcție	Descriere execuție funcție
abs(x)	Valoarea absolută (modul) a unui număr (întreg sau real)
sqrt(x)	Determină valoarea $\sqrt{x}$, ca număr real (x≥0)
exp(x)	Determină valoarea e^x, ca număr real
pow(x,p)	Determină valoarea x^p, ca număr real
log(x)	Determină valoarea ln(x), ca număr real
log10(x)	Determină valoarea lg(x), ca număr real
floor(x)	Determină cel mai mare întreg mai mic sau egal cu x
ceil(x)	Determină cel mai mic întreg mai mare sau egal cu x

Probleme propuse

1. Scrie în limbajul C++ definiția completă a subprogramului **suma_cifre** care primește prin parametrul **n** un număr natural cu cel mult nouă cifre și returnează suma factorialelor cifrelor numărului n. De exemplu, dacă n = 231, subprogramul va returna valoarea 2!+3!+1!=9.
2. Scrie în limbajul C++ definiția completă a subprogramului **baza** care primește prin parametrul **n** un număr natural cu cel mult nouă cifre, primește prin parametrul **b** un număr natural ($2 \le b \le 9$). Subprogramul returnează 1, dacă numărul n reprezintă un număr în baza b și returnează 0 în caz contrar.
3. Scrie în limbajul C++ definiția completă a subprogramului **palindrom** care primește prin parametrul **n** un număr natural cu cel mult nouă cifre. Subprogramul returnează 1, dacă numărul n reprezintă un număr palindrom și returnează 0 în caz contrar.
4. Un număr natural **n** format din cel puțin 5 cifre și cel mult nouă cifre se numește 2– palindrom dacă numărul obținut după eliminarea primelor două cifre și a ultimelor două cifre din numărul n este palindrom. Scrie în limbajul C++ definiția completă a subprogramului **dpalindrom**, care primește două numere naturale **a**, **b** ($10^5 \le a \le b \le 10^9$) și afișează toate numerele 2– palindrom ce aparțin intervalului [a,b]. Funcția dpalindrom va utiliza apeluri utile ale funcției **palindrom** de la problema 3.
5. Subprogramul **numere** primește prin parametrul **n** un număr natural ($2 \le n \le 30$) și prin parametrul **v** un tablou unidimensional format din n elemente, numere naturale formate din cel puțin două cifre și cel mult șase cifre. Subprogramul furnizează prin parametrul **suma** valoarea sumei acelor elemente din tabloul v, pentru care prima cifră este egală cu suma celorlalte sau furnizează valoarea –1 dacă nu există astfel de numere în tablou. Scrie în limbajul C++ definiția completă a subprogramului **numere.** De exemplu, dacă n = 5, v = (125, 624, 44, 2110, 23), atunci suma = 624 + 44 + 2110 = 2778.
6. Se consideră un tablou unidimensional **x,** având **n** elemente, numere naturale ($1 \le n \le 100$) și un număr natural **y** ($0 < y \le 10^9$). Scrie în limbajul C++ definiția completă a subprogramului

prime, care primește ca parametri numerele n, y, tabloul x și furnizează prin parametrul **p** un tablou unidimensional, ce va conține, elementele din tabloul x care sunt prime cu numărul y, iar prin parametrul **np** furnizează numărul de elemente din tabloul p. Astfel, dacă n = 5, y = 3, x = (9, 11, 2, 33 ,7) atunci se va obține p = (11, 2, 7) și np = 3. Utilizează apeluri utile ale subprogramului **cmmdc**, definit mai sus.

7. Se consideră două tablouri unidimensionale **x** și **y**, având fiecare câte **n** elemente ($1 \le n \le 50$). Fiecare tablou conține numere întregi, dispuse în ordine strict crescătoare. Scrie în limbajul C++ definiția completă a subprogramului **egale**, care primește ca parametri numărul **n**, tablourile **x, y** și care returnează prin parametrul **eg** numărul de elemente egale din cele două tablouri. Se va utiliza un algoritm eficient din punct de vedere al timpului de execuție.
8. Scrie în limbajul C++ definiția completă a subprogramului **interschimbare** care primește prin parametrii **n, m, linie1, linie2** trei numere naturale ($2 \le n,m \le 40$, $1 \le$ linie1, linie2 $\le n$), primește prin parametrul **a** un tablou bidimensional de tip întreg, având **n** linii și **m** coloane și interschimbă elementele din cele două linii: linie1 și linie2.
9. Se consideră două puncte A și B definite în planul xOy, având coordonate numere reale. Scrie în limbajul C++ definiția completă a subprogramului **distanta** care primește prin doi parametri punctele A și B și returnează valoarea distanței dintre ele.

Funcții recursive

Teorie

Recursivitatea este o tehnică de programare care se implementează cu ajutorul funcțiilor. O funcție este recursivă dacă se autoapelează. Autoapelul se poate face în interiorul funcției, în acest caz avem **recursivitate directă** sau prin intermediul altei funcții în acest caz avem **recursivitate indirectă**.
O funcție recursivă este bine definită dacă există cazuri elementare care se rezolvă direct (condiție de oprire), iar cazurile neelementare converg către un caz elementar.

Schema generală a unei funcții recursive

```
int f(tip n)
{
  if(caz_elementar)return v;
  else
  {
     Secv_istructiuni1;
     return f(.....);
     Secv_istructiuni2;    }
}
```

```
void f(tip n)
{
  if(!cond_oprire)
  {
     Secv_istructiuni1;
     f(.....);
     Secv_istructiuni2;
  }
}
```

La apelul unei funcții recursive la urcare în stivă se execută `Secv_istructiuni1`, până se ajunge la cazul elementar (condiția de oprire), iar la coborâre în stivă se execută `Secv_istructiuni2`, aceste secvențe de instrucțiuni pot fi vide.

EXEMPLE DE FUNCȚII RECURSIVE

a) Funcții recursive de tip int

- **Suma cifrelor unui număr natural primit ca parametru**

```
int sumcif(int n)
{ if(n<=9)   //caz elementar
     return n;
 else
     return sumcif(n/10)+n%10;
}
```

Structura stivei la apelul `sc(3254)` este:

sc(3)	= 3	↓
sc(32)	= sc(3) + 2 ↑	3 + 2 ↓
sc(325)	= sc(32) + 5 ↑	3 + 2 + 5 ↓
sc(3254)	= sc(325) + 4 ↑	3 + 2 + 5 + 4 ↓

Valoarea returnată va fi 14 = 3 + 2 + 5 + 4

- **Numărul de cifre al unui număr natural primit ca parametru**

```
int nrcif(int n)
{ if(n<=9)   //caz elementar
    return 1;
 else
    return sumcif(n/10)+1;
}
```

Exercițiu: Realizează structura stivei după modelul anterior și scrie ce va returna funcția la apelul nrcif(3254).

- **Partea întreagă a rădăcinii pătrate a unui număr natural n. Funcția se va apela într-o expresie prin `f(0, n + 1, n)`.**

```
int f(int a, int b, int n)
{
            int m = (a + b) / 2;
            if(a==b-1)return a;
            else if (m * m <=n)
            return f(m,b, n);
                 else
return f(a, m, n);
}
```

Exercițiu: Realizează structura stivei după modelul anterior și scrie ce va returna funcția la apelul f(0, 41,40).

- **Șirul lui Fibonacci. Se consideră șirul Fibonacci, definit astfel `f1=1`, `f2=1`, $f_n = f_{n-1} + f_{n-2}$, dacă n>2. Să se determine al n-lea termen al șirului.**

```
int fibonacci(int n)
{
  if(n==1||n==2)return 1;
  else return fibonacci(n-1)+fibonacci(n-2);
}
```

Exemplu: La apelul fibonacci(5) funcția va returna 5

b) Funcții recursive de tip void

- **Afișarea cifrelor unui număr natural, în ordine de la dreapta la stânga urmate de cifrele în ordine de la stânga la dreapta**

```
void afisare(int n)
{
    if(n)
    {
        cout<<n%10<<' ,;
        afisare(n/10);
        cout<<n%10<<' ,;
    }
}
```

Structura stivei la apelul `afisare(1234)` este:

Stiva	Ecran – urcare în stivă	Ecran – coborâre în stivă
afisare(0)	Condiția oprire	↓
afisare(1)	4 3 2 1 ↑	4 3 2 1 1 ↓
afisare(12)	4 3 2 ↑	4 3 2 1 1 2 ↓
afisare(123)	4 3 ↑	4 3 2 1 1 2 3 ↓
afisare(1234)	4 ↑	4 3 2 1 1 2 3 4 ↓

În urma apelului `afisare(1234)` pe ecran se va afișa `4 3 2 1 1 2 3 4.`

- **Transformarea unui număr din baza 10 în baza b (2≤b≤9)**

```
void conversie(int a, int b)
{
    if(a)
    {
        conversie(a/b,b);
        cout<<a%b;
    }
}
```

Exercițiu: Realizează structura stivei după modelul anterior și scrie ce se va afișa la apelul `conversie(120,2)`.

- **Cifra maximă a unui număr natural primit ca parametru**

```
void cifmax(int n, int & cm)
{
 if(n<=9)    //caz elementar
    cm=n;
 else
  {     cifmax(n/10,cm);
if(n%10>cm)cm=n%10;
}
}
```

Exercițiu: Realizează structura stivei după modelul anterior și scrie ce se va afișa la apelul `cifmax(4753,cm)`.

PROBLEME PROPUSE

1. Se consideră următoarea funcție recursivă:

```
int f(int a, int b)
{   if(b==0)
        return a;
    else
    return f(b,a%b);
}
```

Care este efectul funcției și ce returnează funcția la apelul f(120, 80)?

2. Se consideră următoarea funcție recursivă.

```
int f(int a, int b)
{   if(a==b)
        return a;
    else
   if(a>b)return f(a-b,b);
    return f(a,b-a);
}
```

Care este efectul funcției și ce returnează funcția la apelul f(120, 80)?

3. Se consideră următoarea funcție recursivă:

```
int f(int a, int b)
{   if(b==0)
return 1;
    else
return a*f(a,b-1);
}
```

Care este efectul funcției și ce returnează funcția la apelul f(2, 10)?

4. Se consideră următoarea funcție recursivă:

```
int f(int a, int b)
{   if(b==0)
        return a;
    else
  {    int x;
       x=f(a,b/2);
if(b%2==0)
  return x*x;
else return a*x*x;
}
}
```

Care este efectul funcției și ce returnează funcția la apelul f(2, 10)?

5. Se consideră următoarea funcție recursivă:

```
void f(int a, int p)
{    if(a)
  {
if(a%2==1)
  cout<<p<<' ';
f(a/2,p*2);
}
}
```

Care este efectul funcției și ce returnează funcția la apelul f(45,1)?

6. Se consideră următoarea funcție recursivă:

```
void f(int a)
{    if(a)
  {
f(a/16);
if(a%16<10)cout<<a%16;
else cout<<(char)('A'+(a%16-10));
}
}
```

Care este efectul funcției și ce returnează funcția la apelul f(198)?

7. Se consideră funcția recursivă de mai jos. Ce se afișează la apelul f(4)?

```
void f(int n)
{  if(n==1)cout<<1<<' ,;
   else
   {
       f(n-1);
       cout<<n<<' ,;
       f(n-1);
   }
}
```

8. Se consideră următoarea funcție recursivă:

```
int f(int n, int k)
{ if (k == 0 || n == k) return 1;
  else return f(n - 1, k - 1) + f(n - 1, k);
}
```

Care este efectul funcției și ce returnează funcția la apelul f(5,3)?

9. Se consideră funcția recursivă de mai jos. Ce se afișează la apelul f(1, 3)?

```
void f(int a, int b)
{
    for(int i=a; i<=b; i++)
    {
        cout<<i<<' ,;
        f(i+1,b);
    }
}
```

CAPITOLUL 10

Metoda backtracking. Elemente de combinatorică

Teorie

Metoda Backtracking este o metodă de programare care se aplică problemelor de generare în care soluția se poate reprezenta sub forma unui vector x = (x1, x2, ... xn) ∈ S1 x S2 x ... x Sn unde mulțimile S1, S2, ... Sn sunt mulțimi finite și nevide. Pentru fiecare problemă sunt date anumite relații între componentele vectorului x, numite **condiții interne**. Produsul cartezian S1 x S2 x ... x Sn reprezintă **spațiul soluțiilor posibile**. Elementele vectorului soluție se generează în ordine, x1, x2, ..., xn. Valoarea atribuită lui x_k trebuie să îndeplinească anumite condiții în raport cu valorile deja generate numite **condiții de continuare**. Condițiile de continuare se deduc din **condițiile interne**.

Algoritmul general al metodei poate fi implementat recursiv sau iterativ. Înainte de a scrie algoritmul de rezolvare al unei probleme prin metoda backtracking trebuie stabilite detalii cu privire la:

- vectorul soluție (numărul de componente și semnificația componentelor);
- mulțimea valorilor posibile pentru fiecare componentă;
- condițiile interne;
- condițiile de continuare (sunt verificate în funcția `int valid(int k)`);
- condiția ca vectorul generat să fie soluție.

Odată stabilite aceste detalii putem utiliza un algoritm general de bactracking, care poate fi particularizat în funcție de problema ce trebuie rezolvată.
Dacă o componentă oarecare x[k], ia valori consecutive între două limite cunoscute prim[k] și ultim[k] și numărul de componente din vectorul soluție este n putem utiliza următorul algoritm de tip backtracking:

```
void back(int k)
{    int v;
if (k==n+1)
afisare_sol();
else
{
          for(v=prim[k];v<=ultim[k];v++)
          {
               x[k]=v;
               if (valid(k))back(k+1);
          }
   }
   }
```


Elemente de combinatorică

Generarea permutărilor

Fie *n* un număr natural, să se genereze toate permutările mulțimii {1, 2, ..., n} ($P_n = n!$).

1) $x[i] \in \{1, 2, .., n\}$
2) Elementele vectorului soluție trebuie să fie distincte, $x[i] \neq x[j]$, oricare $i \neq j$, la pasul *k* vom compara elementul $x[k]$ cu $x[i]$, unde $i = 1, k-1$.
3) Obținem soluție dacă am generat cele *n* elemente din vectorul soluție.

Exemplu: Să se genereze permutările mulțimii {1, 2, 3}.

```
1 2 3; 1 3 2; 2 1 3; 2 3 1; 3 1 2; 3 2 1
```

Generarea combinărilor

Fie *m* și *n* numere naturale, $m \leq n$. Să se genereze submulțimile cu *m* elemente ale mulțimii {1, 2, ..., n}, submulțimi în care nu contează ordinea. Numărul submulțimilor cu această proprietate este: $C_n^m = \frac{n!}{(n-m)! \cdot m!}$.

1) $x[i] \in \{1, 2, ..., n\}$
2) Elementele vectorului soluție trebuie să fie distincte, **pentru a nu genera de mai multe ori aceeași submulțime vom genera elementele vectorului soluție în ordine strict crescătoare.** În acest caz x[k] ∈ {x[k-1]+1, ..., n-m+k}, k>=1, în acest caz orice valoare posibilă a unui element este validă (nu vom avea condiții de continuare).
3) Obținem soluție dacă am generat cele m elemente ale vectorului soluție.

Exemplu: Să se genereze submulțimile cu 3 elemente ale mulțimii {1, 2, ..., 5}, submulțimi în care nu contează ordinea. (10 soluții)

```
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
```

Generarea aranjamentelor (generarea funcțiilor injective)

Fie m și n numere naturale, m≤n. Să se genereze submulțimile cu m elemente ale mulțimii {1, 2, ..., n}, submulțimi în care contează ordinea. Numărul submulțimilor cu această proprietate este: $A_n^m = \frac{n!}{(n-m)!}$.

Singura diferență între generarea aranjamentelor și a permutărilor constă în dimensiunea vectorului soluție. Dacă la permutări obținem soluție când am generat n elemente, la aranjamente obținem soluție când am generat m elemente. Aranjamente de n luate câte m reprezintă numărul funcțiilor injective de forma: f:{1,2,...,m} →{1,2,...,n}.

Exemplu: Să se genereze submulțimile cu 3 elemente ale mulțimii {1, 2, ..., 5}, submulțimi în care contează ordinea. (60 de soluții)

```
1 2 3; 1 2 4; 1 2 5; 1 3 2; 1 3 4; 1 3 5; 1 4 2; 1 4 3; 1 4 5;
1 5 2; 1 5 3; 1 5 4; .....; 5 1 2; 5 1 3; 5 1 4; 5 2 1; 5 2 3;
5 2 4; 5 3 1; 5 3 2; 5 3 4; 5 4 1; 5 4 2; 5 4 3.
```

Generarea produsului cartezian

Se consideră n mulțimi de cardinale $p_1, p_2, ..., p_n$. Se cere să se determine elementele produsului cartezian M_1 x M_2 x ...x M_n. Numărul de elemente din vectorul soluție este card(M_1)* card(M_2)*...* card(M_n).

1) $x[i] \in Mi$
2) Orice valoare posibilă a unui element este validă (nu vom avea condiții de continuare).
3) Obținem soluție dacă am generat cele n elemente din vectorul soluție.

Exemplu: Să se genereze elementele produsului cartezian {1,2} x {*a*,*b*} x {1,2}, (8 soluții)

```
1 a 1; 1 a 2; 1 b 1; 1 b 2; 2 a 1; 2 a 2; 2 b 1; 2 b 2
```

Generarea partițiilor unei mulțimi

Fie $n \in N^*$. Să se genereze partițiile mulțimii `A = {1,2,..., n}`.

Definiție: O partiție a mulțimii A este un set de submulțimi ale lui A cu următoarele proprietăți:

a) $A_i \neq \emptyset$, $\forall i \in \{1,2,...,k\}$

b) $A_i \cap A_j = \emptyset$, $\forall i \neq j$, $i,j \in \{1, 2, ...,k\}$

c) $A = A_1 \cup A_2 \cup A_3 ... \cup A_k$

Pentru a genera toate partițiile unei mulțimi trebuie să plasăm fiecare element din mulțime într-o submulțime, elementul face parte dintr-o submulțime deja creată sau va fi primul element dintr-o nouă submulțime. Putem atașa unei partiții un vector V cu proprietatea:

`V[i] ∈ {V[1], V[2],..., V[i-1]}` sau `V[i] = max{V[j]/ j=1,i-1}+1`

Exemplu: Să se genereze partițiile mulțimii `{1,2,3}`.

```
{1,2,3}    vectorul atașat 1 1 1
{1,2} {3}  vectorul atașat 1 1 2
{1,3} {2}  vectorul atașat 1 2 1
{1}{2,3}   vectorul atașat 1 2 2
{1}{2}{3}  vectorul atașat 1 2 3
```

Generarea partițiilor unui număr

Fie $n \in N^*$. Să se determine toate partițiile numărului natural n.

Definiție: Numim partiție a unui număr natural n, nenul un set de numere naturale nenule cu proprietatea că `n = x[1] + x[2] + ... + x[k]`

1) `x[i] ∈ {1,..., n}`
2) Pentru a nu genera de mai multe ori aceeași partiție vom genera elementele partiției în ordine crescătoare, adică `x[i + 1] ≥ x[i]`, `∀ i ∈ {1,2,..., k}` astfel că vom inițializa `x[0]` cu `1`.
3) `n = x[1] + x[2] + ... + x[k]`

Exemplu: Să se genereze partițiile numărului `n = 5`.

```
1 1 1 1 1
1 1 1 2
1 1 3
1 2 2
1 4
2 3
5
```

Probleme propuse

1. Un program C++ determină numărul modalităților de a scrie un număr natural **n** ca sumă de cel puțin două numere naturale distincte. Două soluții sunt distincte dacă diferă prin cel puțin un număr. De exemplu pentru **n** = 8 se va afișa 5, (soluțiile sunt 1 + 2 + 5, 1 + 3 + 4, 1 + 7, 2 + 6, 3 + 5). Care este valoarea ce se va afișa dacă **n** = 12?

2. Un atelier de croitorie trebuie să coase steaguri tricolore având pânză de culorile alb, albastru, galben, portocaliu, roșu și verde. Două steaguri sunt diferite dacă ordinea culorilor este diferită; într-un steag culorile nu se repetă. Primele trei steaguri construite sunt: alb, albastru, galben; alb, albastru, portocaliu; alb, albastru, roșu. În câte moduri poate combina cele trei culori (numărul de steaguri distincte care se pot coase) și care este ultimul steag obținut?

3. Dirigintele unei clase a XII-a, trebuie să aleagă o echipă formată din trei elevi care să participe la o conferință pe teme informatice. Colegii i-au propus pe Maria, Radu, Robert, Teodora și Vlad. Câte posibilități de alegere are dirigintele, care este algoritmul de generare și care este ultima echipă formată?

4. Pentru generarea lexicografică a submulțimilor unei mulțimi `S = {1,2,..., n}` se utilizează un algoritm de tip backtracking, astfel pentru **n** = 3 se generează submulțimile {}, {1}, {1, 2}, {1, 2, 3}, {1, 3}, {2}, {2, 3}, {3}. Utilizând același algoritm pentru **n** = 4, care este a cincea soluție generată?

5. Se utilizează algoritmul de generare al produsului cartezian pentru generarea produsului cartezian `A x B x C` al mulțimilor $A = \{a, b\}$, B $\{b\}$, $C = \{c, d, e\}$. Care este cel de-al patrulea element generat?

6. Utilizând metoda backtracking se generează toate numerele palindrom formate din 4 cifre din mulțimea {1, 2, 3}. Numerele generate sunt 1111, 1221, 1331, 2112, 2222, 2332, 3113, 3223, 3333. Dacă se folosește aceeași metodă pentru generarea palindroamelor de 4 cifre din mulțimea `{1,2,3,4,5,6,7,8,9}`, să se determine câte numere palindrom se vor genera și câte dintre aceste numere sunt pare?

7. Se utilizează metoda backtracking pentru generarea permutărilor de 5 obiecte, primele 4 permutări sunt 5 4 3 2 1; 5 4 3 1 2; 5 4 2 3 1; 5 4 2 1 3. Care este a şaptea permutare?

8. Se generează, prin metoda backtracking, toate partiţiile mulţimii $A = \{1,2,3\}$ obţinându-se următoarele soluţii: {1,2,3};{1,2}{3};{1,3}{2};{1}{2,3};{1}{2}{3}. Se observă că dintre acestea, a doua soluţie e alcătuită din exact două submulţimi. Dacă se foloseşte aceeaşi metodă pentru a genera partiţiile mulţimii {1,2,3,4} stabiliţi câte dintre soluţiile generate vor fi alcătuite din exact două submulţimi.

9. Se generează cele 10 submulţimi cu 3 elemente ale mulţimii {1, 2, 3, 4, 5} în care nu contează ordinea: 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 două dintre submulţimi conţin secvenţa 2 4 (elementele sunt pe poziţii consecutive). Dacă se generează submulţimi cu 4 elemente ale mulţimii {1, 2, 3, 4, 5, 6} în care nu contează ordinea, câte submulţimi conţin secvenţa 2 4?

CAPITOLUL 11

Elemente de teoria grafurilor

Teorie

1. Grafuri neorientate

- terminologie (nod/vârf, muchie, adiacenţă, incidenţă, grad, lanţ, ciclu, elementar, lungime, subgraf, graf parţial, graf regulat/complet/bipartit/aciclic/hamiltonian/eulerian/conex, componentă conexă)
- formule/proprietăţi (suma gradelor vârfurilor, numărul de grafuri neorientate, numărul de grafuri parţiale, numărul maxim de muchii)
- metode de reprezentare în memorie (matrice de adiacenţă, liste de adiacenţă, lista de muchii)

Definiţie: Un ***graf*** este o pereche ordonată de mulţimi, notată $G = (X,U)$, unde:
$X = \{x | x \in X\}$ este ***mulţimea nodurilor*** (vârfurilor), iar $U = \{(x,y) | x,y \in X\}$, ***mulţimea muchiilor***.
nod/vârf = element al mulţimii X ce poate fi reprezentat în plan printr-un punct (cerc etc.), eventual numerotat
muchie = pereche neordonată de noduri ce poate fi reprezentată în plan printr-un segment de dreaptă/arc
adiacenţă = proprietate a două noduri de a fi unite prin muchie; dacă $[x, y] \in U$, spunem că nodurile x şi y sunt adiacente
incidenţă = proprietatea unei muchii de a uni două noduri; dacă $[x, y] \in U$, spunem că muchia este incidentă cu nodurile x şi y
gradul nodului x = numărul de muchii incidente cu nodul x, notat cu $d(x)$
nod izolat = nod cu gradul 0; `d(x)=0`
nod terminal = nod cu gradul 1; `d(x)=1`
`d(x)` $\leq$ `n-1,` $\forall$ `x` $\in$ `X, |X|=n`
Propoziţie: În orice graf neorientat cu n noduri şi m muchii, are loc egalitatea:
$2*m = d(x_1)+d(x_2)+...+d(x_n)$
(Suma gradelor vârfurilor este dublul numărului de muchii)
Consecinţă: În orice G există un număr PAR de vârfuri de graf IMPAR.

lanţ = succesiune de noduri cu proprietatea că oricare două noduri consecutive din lanţ sunt adiacente
lanţ compus = lanţ în care muchiile se pot repeta
lanţ simplu = lanţ în care fiecare muchie apare o singură dată, dar nodurile se pot repeta
lanţ elementar = lanţ în care nodurile sunt distincte

- Numărul de subgrafuri complete este 19 (8 subgrafuri cu câte un vârf, 9 subgrafuri cu câte 2 vârfuri adiacente și 2 subgrafuri complete cu 3 vârfuri).
- Numărul de grafuri parțiale ale grafului din Fig. 1, este $2^9 = 512$ (graful are 9 muchii și orice combinație a acestora poate fi aleasă în graful parțial).

Metode de reprezentare a grafurilor neorientate în memorie

Matricea de adiacență

- $a \in M_{n \times n}$ $a_{xy} = \begin{cases} 1, \textit{dacă nodurile } x \textit{ și } y \textit{ sunt adiacente} \\ 0, \textit{altfel} \end{cases}$
- matricea este simetrică față de diagonala principală
- gradul unui nod, `d(x)` = numărul de valori egale cu 1 de pe linia/coloana *x*

Listele de adiacență

`Li = {j ∈ X/∃[i,j] ∈ U}, i≤n`

Lista de muchii

`t ∈ M`$_{2 \times m}$, unde *m* = numărul de muchii din graf
`t1, k` și `t2, k` = extremitățile muchiei *k*

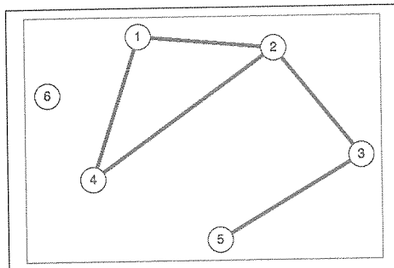


Fig. 2

Graf neorientat cu 6 noduri

$$a = \begin{bmatrix} 0&1&0&1&0&0 \\ 1&0&1&1&0&0 \\ 0&1&0&0&1&0 \\ 1&1&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&0&0&0 \end{bmatrix}$$

Matricea de adiacență

L1: 2,4
L2: 1,3,4
L3: 2,5
L4: 1,2
L5: 3
L6: -

Listele de adiacență

$$t = \begin{bmatrix} 2&4&4&3&5 \\ 1&1&2&2&3 \end{bmatrix}$$

Lista de muchii

PROBLEME PROPUSE

1. Rezolvă următoarele cerințe pe graful din figura alăturată.
   a) Enumeră nodurile de grad maxim.
   b) Scrie listele de adiacență ale grafului dat.
   c) Scrie un ciclu elementar de lungime maximă.
   d) Care este numărul minim de muchii de eliminat astfel încât graful rezultat să fie eulerian? Precizează o posibilă soluție.

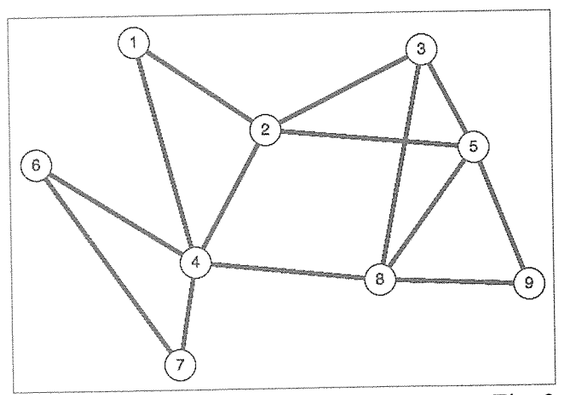


Fig. 3

e) Câte subgrafuri complete, fără vârfuri izolate conține graful dat?
f) Scrie linia corespunzătoare nodului **4** din matricea de adiacență.
g) Câte muchii trebuie adăugate grafului astfel încât să devină graf complet?
h) Care sunt componentele conexe ale subgrafului indus de vârfurile de grad impar?

2. Fie **G** un graf neorientat cu **30** de vârfuri reprezentat printr-o matrice de adiacență cu **24** de valori nenule.

   Graful poate fi format din cel puțin ..... și cel mult ..... componente conexe.

3. Fie **G** un graf cu **25** de muchii, fără noduri izolate. Numărul maxim de noduri din graf este.....

4. Câte grafuri neorientate cu **10** noduri se pot forma astfel încât nodul **1** să fie izolat?

5. Care este numărul minim de noduri dintr-un graf neorientat cu **18** muchii, fără vârfuri izolate, format din trei componente conexe? Dar numărul maxim de noduri?

6. Câte subgrafuri euleriene cu **3** vârfuri, conține un graf neorientat, complet, cu **8** vârfuri?

7. Un graf neorientat conex cu **20** de vârfuri are exact două cicluri elementare. Știind că cele două cicluri elementare nu au noduri în comun și au lungimile **6**, respectiv **8**, care este numărul total de muchii din **G**?

8. Într-un graf neorientat conex, *distanța* dintre două noduri **u** și **v** se definește ca fiind lungimea minimă a unui lanț de la **u** la **v**. *Puterea* unui nod **v** se definește ca fiind suma distanțelor de la **v** la celelalte noduri. Care este valoarea maximă pe care o poate avea *puterea* unui nod într-un graf conex neorientat cu **n>1** noduri?

9. Fie graful din imaginea alăturată. Este bipartit? Dacă da, scrie mulțimile unei posibile partiții și precizează câte muchii trebuie adăugate grafului pentru a deveni bipartit complet.

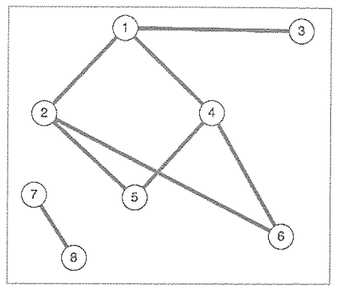


Fig. 4

Teorie

2. Grafuri orientate

- terminologie (nod/vârf, arc, adiacență, incidență, grad intern/grad extern, drum/drum elementar, circuit/circuit elementar, lungime, subgraf, graf parțial, graf plin/complet/turneu/tare conex, componentă tare conexă)
- formule/proprietăți (suma gradelor interne/externe vârfurilor, numărul de grafuri orientate, numărul de grafuri complete, numărul de grafuri turneu, numărul maxim de arce)
- metode de reprezentare în memorie (matrice de adiacență, liste de adiacență)

Definiție: Un ***graf orientat*** este o pereche ordonată de mulțimi, notată $G = (X,U)$, unde $X = \{x|x \in X\}$ este ***mulțimea nodurilor*** (vârfurilor), iar $U = \{(x,y)|\ x,y \in X\}$, ***mulțimea arcelor***.

nod/vârf = element al mulțimii X; poate fi reprezentat în plan printr-un punct (cerc etc.), eventual numerotat
arc = pereche ordonată de noduri; poate fi reprezentată în plan printr-o săgeată orientată
adiacență = proprietatea a două noduri de a fi unite prin arc; dacă $(x,y) \in U$, spunem că nodurile x și y sunt adiacente
incidență = proprietatea unui arc de a uni două noduri; dacă $(x,y) \in U$, spunem că arcul este incident cu nodul x
gradul intern al nodului x = $d^-(x)$ = numărul de arce care intră în nodul x (x = extremitate finală)
gradul extern al nodului x = $d^+(x)$ = numărul de arce care ies din nodul x (x = extremitate inițială)

nod izolat = nod cu gradul intern și extern 0; $d^-(x) = d^+(x) = 0$
$d^+(x) \leq n-1$, $d^-(x) \leq n-1$

Propoziție: În orice graf orientat cu n noduri și m arce, are loc egalitatea
Suma gradelor interioare = suma gradelor exterioare = numărul de arce.

drum = succesiune de noduri cu proprietatea că oricare două noduri consecutive sunt adiacente (arcele pastrează aceeași orientare)

drum elementar = drum în care nodurile sunt distincte

circuit = drum în care primul nod coincide cu ultimul

circuit elementar = circuit în care nodurile sunt distincte, cu excepția primului și a ultimului nod

lungimea unui drum/circuit = numărul de arce din care este format

graf parțial = graf care se obține din graful inițial prin eliminarea unor arce, nu și a nodurilor
subgraf = graf care se obține din graful inițial prin eliminarea unor noduri și a tuturor arcelor care au o extremitate în nodurile eliminate; nu pot fi eliminate alte arce decât cele cu extremități în nodurile eliminate

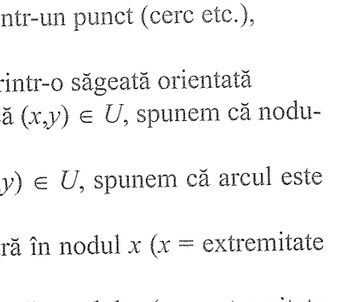


Fig. 5

Pe graful din Fig. 5, punem în evidență noțiunile prezentate anterior:

- nodurile 1 și 4 sunt adiacente
- nodurile 1 și 6 nu sunt adiacente
- $d^+(4) = 2$; $d^-(4) = 3$; $d^+(3) = 0$; $d^-(3) = 3$
- $L = [1, 2, 4, 7, 6, 4, 3]$ este drum neelementar de lungime 6 (se repetă nodul 4 și este format din 6 arce)
- $L = [5, 1, 2, 4, 7, 6]$ este drum elementar de lungime 5 (nu se repetă noduri și este format din 5 arce)
- $C = [5, 1, 2, 1, 5]$ este circuit neelementar de lungime 4 (5 este nodul inițial și final, iar nodul 1 se repetă; circuitul este format din 4 arce)

- $C = [4, 7, 6, 4]$ în loc de $C = [1, 2, 4, 1]$ este circuit elementar de lungime 3 (circuitul este format din 3 arce)

Tipuri particulare de grafuri orientate

graf plin = un graf orientat în care $\forall\, x \neq y$, cu $x, y \in X$, $\exists\, (x, y)$ și $\exists\, (y, x)$ (există arc dus-întors)

Numărul de arce într-un graf plin cu n noduri este $n(n-1)$.

Numărul de grafuri orientate cu n noduri este $2^{n(n-1)} = 4^{C_n^2}$.

graf complet orientat = un graf orientat în care $\forall\, x \neq y$, cu $x, y \in X$, $\exists\, (x, y)$ sau $\exists\, (y, x)$ sau $\exists\, (x, y)$ și $\exists\, (y, x)$

Graful complet orientat **nu** este unic, iar numărul de arce m respectă relația

$$\frac{n(n-1)}{2} \leq m \leq n(n-1)$$

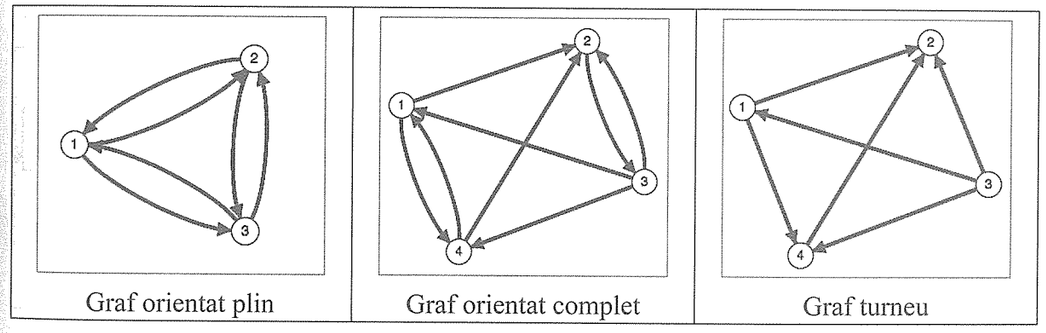
Numărul de grafuri orientate **complete** cu n noduri este = $3^{C_n^2} = 3^{\frac{n(n-1)}{2}}$.

graf turneu = un graf orientat în care $\forall\, x \neq y$, cu $x, y \in X$, există exact un arc între ele.

Graful turneu **nu** este unic, iar numărul de arce m respectă relația:

$$m = \frac{n(n-1)}{2}$$

Numărul de grafuri **turneu** cu n noduri este = $2^{C_n^2} = 2^{\frac{n(n-1)}{2}}$



Graf orientat plin | Graf orientat complet | Graf turneu

Fig. 6

graf orientat tare conex = oricare ar fi două noduri distincte x și y, există drum dus-întors de la x la y (există un circuit, nu neapărat elementar, care trece prin toate arcele grafului orientat)

componentă tare conexă a unui graf orientat = un subgraf tare conex și maximal în raport cu această proprietate (nu există drum dus-întors între un nod din subgraf și un nod care nu aparține subgrafului)

Obs: Un nod izolat constituie o componentă tare conexă.

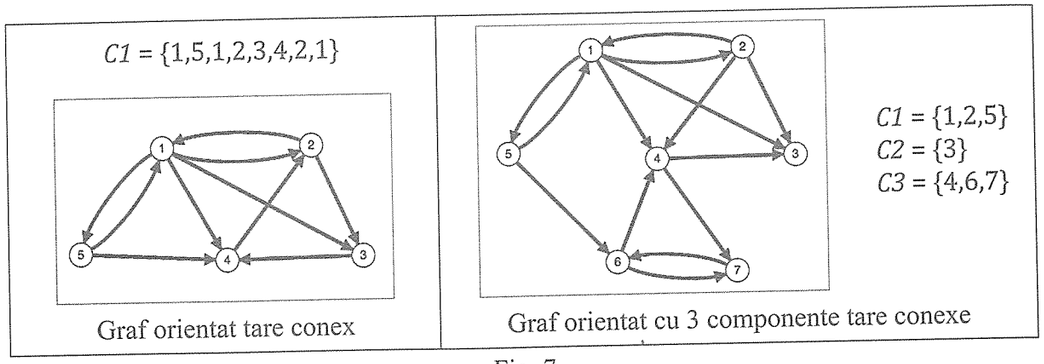


Fig. 7

Metode de reprezentare a grafurilor orientate în memorie

a) Matricea de adiacență

- $a \in M_{n \times n}$ $a_{xy} = \begin{cases} 1, \textit{dacă există arcul } (x,y) \in U \\ 0, \textit{altfel} \end{cases}$
- matricea de adiacență nu este simetrică față de diagonala principală
- gradul exterior al nodului x, $d^+(x)$ = numărul de valori egale cu 1 de pe linia x
- gradul interior al nodului x, $d^-(x)$ = numărul de valori egale cu 1 de pe coloana x

b) Lista de arce

`t∈M2xm`, unde `m` = numărul de arce din graf
`t1,k` și `t2,k` = extremitățile arcului k

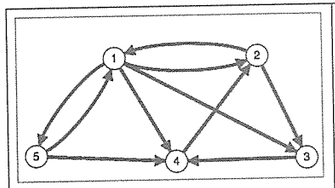


Fig. 8

$$a = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

L1: 2,3,4,5
L2: 1,3
L3: 4
L4: 2
L5: 1,4

Matricea de adiacență
Listele de adiacență

$$t = \begin{bmatrix} 1 & 2 & 1 & 2 & 3 & 4 & 1 & 1 & 5 & 5 \\ 2 & 1 & 3 & 3 & 4 & 2 & 4 & 5 & 1 & 4 \end{bmatrix}$$

Lista de arce

c) Listele de adiacență

$L_x = \{y \in X / (x,y) \in U\}$

PROBLEME PROPUSE

1. Fie **G** un graf orientat cu **30** de vârfuri în care suma gradelor interioare ale acestora este **50**. Graful poate fi format din cel mult ..... vârfuri izolate.
2. Fie **G** un graf orientat cu **20** de arce, fără vârfuri izolate. Numărul minim de vârfuri din graf este..... iar numărul maxim este.....
3. Câte grafuri orientate complete, cu **10** noduri se pot forma?
4. Care este numărul maxim de noduri dintr-un graf orientat cu **18** arce, fără vârfuri izolate, format din trei componente tare conexe?
5. Graful orientat complet **G** cu **10** vârfuri are cel mult ..... arce și cel mult ..... arce.
6. Rezolvă următoarele cerințe pe graful orientat din figura alăturată.
   a) Scrie gradul interior și gradul exterior al lui 2.
   b) Scrie lista de adiacență a nodului 5.
   c) Scrie un drum elementar de lungime maximă de la 1 la 4.
   d) Scrie coloana corespunzătoare nodului 3 din matricea de adiacență.
   e) Scrie nodurile care au gradul interior mai mare decât gradul exterior.
   f) Care sunt componentele tare conexe ale grafului dat?
   g) Care este numărul minim de arce de adăugat pentru ca graful alăturat să devină tare conex?
   h) Scrie circuitele elementare conținute în graful alăturat.

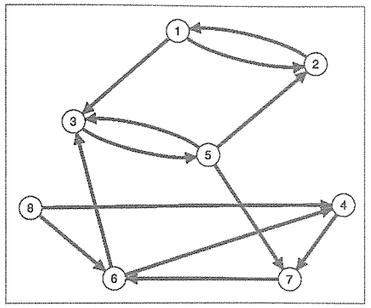


Fig. 9

7. Într-un graf orientat cu **100** de vârfuri există doar arcele **(x, y)**, unde **x<y** și **x, y** au aceeași paritate.
   a) Câte componente tare conexe are graful?
   b) Care este numărul minim de arce de adăugat astfel încât graful să fie tare conex? Scrie arcele unei posibile soluții.
8. Fie graful orientat **G**, în care nodurile sunt etichetate cu cele șapte note muzicale **DO, RE, MI, FA, SOL, LA, SI.** Spunem că vârful **x** domină vârful **y** dacă eticheta lui **x** este mai mare lexicografic decât eticheta lui **y** (de exemplu, **SOL** domină **MI**). Scrie listele de adiacență ale grafului și un drum elementar care trece prin toate vârfurile grafului.
9. Fie graful orientat reprezentat în Fig. 10. Scrie matricea a_{6x6}, în care

$$a_{i,j} = \begin{cases} 1, \textit{dacă există drum de la } i \textit{ la } j \\ 0, \textit{altfel} \end{cases}, \quad 1 \le i, j \le 6$$

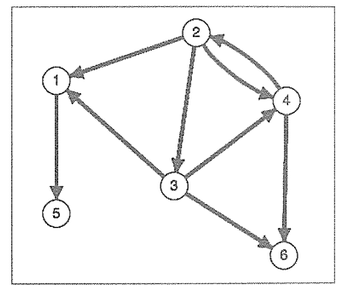


Fig. 10

Teorie

3. Arbori

- terminologie (nod/vârf, muchie, rădăcina, descendent, descendent direct/fiu, ascendent, ascendent direct/părinte, frați, nod terminal, frunză, înălțime, niveluri, arbore degenerat, arbore binar)
- metode de reprezentare în memorie (matrice de adiacență, liste de descendenți, vectori de tați)

Definiție: Un ***arbore*** este un graf conex aciclic.

Teorema de caracterizare
Următoarele afirmații sunt echivalente:
1) A este arbore cu **n** vârfuri.
2) A este graf conex cu **n−1** muchii.
3) A este graf aciclic cu **n−1** muchii.
4) A este graf conex minimal (dacă se elimină o muchie, se distruge conexitatea).
5) A este graf aciclic maximal (dacă se adaugă o muchie, se formează un ciclu).

Proprietate: Oricare ar fi două noduri distincte în arbore, există un lanț elementar **unic** între ele. Într-un graf conex cu n noduri și m muchii, numărul de muchii de eliminat pentru a-l transforma în arbore este **m − n + 1**.

Definiție: Un ***arbore cu rădăcină*** este un arbore în care există un nod special numit ***rădăcină***, iar toate celelalte noduri reprezintă descendenți direcți sau indirecți ai rădăcinii.

descendent al nodului x = nod care se află pe un lanț elementar ce pleacă din **x**, altul decât cel care unește rădăcina de **x**.
fiu/descendent direct al nodului x = descendent al nodului **x**, adiacent cu **x** (nod adiacent cu **x** care **nu** se află pe lanțul care unește rădăcina de nodul **x**)
ascendent al nodului x = nod care se află pe lanțul elementar care unește rădăcina de nodul **x**.
părinte/tată/ascendent direct al nodului x = ascendent al nodului **x** adiacent cu **x**.
frunză/terminal = nod care nu are descendenți (are gradul 1)
arbore degenerat = arbore în care orice nod care nu este terminal, are exact un descendent direct/fiu.
înălțimea arborelui = numărul maxim de muchii al unui lanț de la rădăcină la o frunză
Înălțimea maximă a unui arbore cu rădăcină cu **n** noduri este egală cu **n−1** și se obține în cazul arborelui degenerat.
arbore binar = arbore vid sau arbore în care orice nod are cel mult doi fii, între care se face distincție clară: fiu stâng, fiu drept.

Numărul maxim de noduri într-un arbore binar de înălțime **h** este $2^{h+1} - 1$.

Clase speciale de arbori binari:
- *Arbore binar plin* – are ***k*** niveluri numerotate de la **0** la ***k−1*** și pe fiecare nivel *x* au 2^x noduri. Numărul total de noduri dintr-un arbore binar plin va fi: $n = 2^k - 1 = 2^0 + 2^1 + \ldots + 2^{k-1}$.

Obs. Orice lanț elementar de la rădăcină la un nod din arbore va avea lungimea cel mult $k = [\log_2 n]$.
- *Arbore binar strict* – orice nod care nu este frunză are exact doi fii.
- *Arbore binar complet* – se obține dintr-un arbore binar plin prin eliminarea, în ordine, a unor frunze de pe ultimul nivel, de la dreapta spre stânga.
- *Arbore binar echilibrat* – arbore binar în care, pentru orice nod, numărul de noduri din subarborele său stâng diferă de numărul de noduri din subarborele său drept prin cel mult o unitate.
- *Arbore binar degenerat* – arbore binar în care fiecare nod cu excepția frunzei, are exact un fiu (stâng sau drept). Un arbore degenerat cu **n** vârfuri va avea n niveluri.

Exemplificăm noțiunile de mai sus pe arborele din Fig. 11.

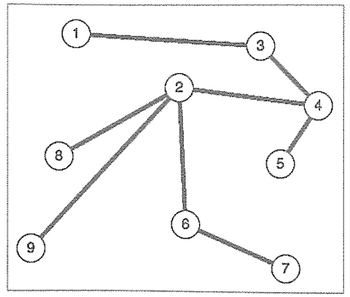


Fig. 11

- Arborele din imagine are 9 noduri și 8 muchii.
- Este aciclic maximal – dacă adăugăm o muchie între orice două noduri neadiacente, se formează un ciclu.
- Lanțul elementar de lungime maximă 5 este $L = [1, 3, 4, 2, 6, 7]$.
- Nodurile 1 și 7 pot fi alese ca rădăcină astfel încât înălțimea să fie maximă (înălțimea = 5).
- Dacă rădăcina este nodul 4 atunci:
  - Nodurile 2, 3, 5 sunt fii/descendenții direcți ai nodului 4.
  - Descendenții nodului 2 sunt: 6, 7, 8, 9.
  - Frații nodului 9 sunt nodurile 8 și 6.
  - Ascendenții nodului 7 sunt 6, 2, 4.
  - Nodurile frunză/terminale sunt 1, 5,7, 8, 9.
  - Înălțimea arborelui este 3.

Metode de reprezentare a arborilor în memorie

1. **Matricea de adiacență – reprezentarea în memorie a arborelui ca și graf neorientat**

$$a \in \mathcal{M}_{n \times n} \quad a_{xy} = \begin{cases} 1, \text{dacă există muchia } (x, y) \in U \\ 0, \text{altfel} \end{cases}$$

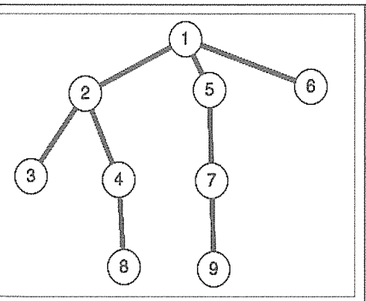
2. **Reprezentare ascendentă pentru arbori cu rădăcină – EFICIENTĂ**

$$tata(n),\ tata_x = \begin{cases} \text{părintele nodului } x, \text{ dacă } x \neq \text{rădăcină} \\ 0, \text{altfel} \end{cases}$$

De exemplu, pentru arborele din Figura 11: rad = 4, tata = (3, 4, 4, 0, 4, 2, 6, 2, 2)

3. **Reprezentare descendentă pentru arbori cu rădăcină** – pentru orice nod care **nu** este frunză reține lista de fii

$$fiu(n),\ fiu_x = \begin{cases} \text{lista fiilor nodului } x, \text{ dacă } x \neq \text{frunza} \\ 0, \text{altfel} \end{cases}$$

 Fig 12	Tata = (0, 1, 2, 2, 1, 1, 5, 4, 7) • Rădăcina este singurul nod cu Tata[radacina]=0 • Este o reprezentare eficientă • Nodurile frunză sunt cele care nu apar în vectorul Tata: (3,8,9,6) • Arborele are 4 niveluri și înălțimea 3 • Gradul nodului 2 este d(2)=3 (are 3 adiacenți: 1,3,4) • Nodul 2 are 2 fii: 3,4	L1: 2, 5, 6 L2: 3, 4 L3: - L4: 8 L5: 7 L6: - L7: 9 L8: - L9: -
Arbore cu rădăcina 1	Vectorul de tați	Liste de fii

Reprezentare stânga-dreapta pentru arbori binari – pentru fiecare nod care **nu** este frunză se reține fiul stâng și fiul drept.

$$stânga_x = \begin{cases} \textit{fiul stâng al lui x}, \text{ dacă există} \\ 0, \textit{altfel} \end{cases},$$

$$dreapta_x = \begin{cases} \textit{fiul drept al lui x}, \text{ dacă } \textit{există} \\ 0, \textit{altfel} \end{cases}$$

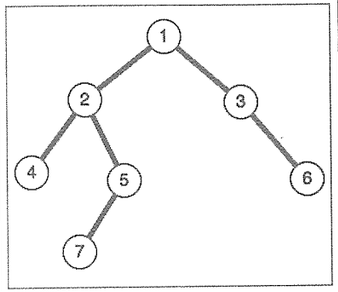


Fig. 13

Arbore cu rădăcina 1

stânga = (2, 4, 0, 0, 7, 0, 0),
dreapta = (3, 5, 6, 0, 0, 0, 0)

- Nodurile frunză sunt (4, 7, 6)
- Arborele are 4 niveluri și înălțimea 3
- Gradul nodului 2 este d(2) = 3 (are 3 adiacenți: 1, 3, 4)
- Nodul 2 are 2 fii: 4, 5

Probleme propuse

1. Rezolvă următoarele cerințe pe arborele cu rădăcină din Fig. 14, considerând că rădăcina este nodul **1**.
   a) Care sunt frunzele arborelui?
   b) Ce înălțime are arborele cu rădăcina **1**?
   c) Ce noduri pot fi alese ca rădăcină astfel încât orice vârf să aibă cel mult doi fii?
   d) Care sunt ascendenții nodului **7**?
   e) Care este lungimea maximă a unui ciclu elementar care se poate forma în arborele dat prin adăugarea unor muchii?

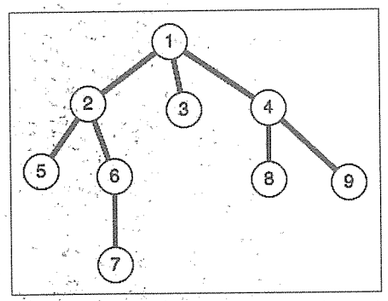


Fig. 14

2. Fie **G** un graf neorientat aciclic, format din trei componente conexe cu câte **4**, **5**, respectiv **7** noduri. Câte muchii conține graful dat?
3. Se consideră un arbore în care fiecare vârf care nu este frunză are gradul **3** iar înălțimea arborelui este **10**. Care este numărul minim de noduri din arbore?
4. Fie arborele cu rădăcină descris prin următorul vector $tata = (6, 5, 5, 2, 0, 3, 3, 3, 8, 7, 7)$
   a) Reprezintă grafic arborele.
   b) Ce noduri pot fi alese ca rădăcină astfel încât înălțimea să fie maximă/minimă?
   c) Pentru nodul **3** ales ca rădăcină, scrieți vectorul *tata*.
   d) Care este lungimea maximă a unui lanț elementar din acest arbore? Câte astfel de lanțuri sunt în arbore?
5. Scrie numărul maxim de muchii de eliminat dintr-un graf complet cu **6** noduri astfel încât să devină arbore.
6. Într-un arbore cu **30** vârfuri, fiecare vârf **n**, cu excepția rădăcinii, are părintele (**tata**), egal cu **[n/2]**. Scrie un lanț elementar care unește nodurile **16** și **20**.
7. Într-un arbore cu înălțimea **8**, fiecare nod care nu este frunză are cel mult doi fii. Numărul de noduri este cel mult..... și cel puțin .....
8. Într-un arbore cu **15** muchii, fiecare nod care nu este frunză are mai mulți fii decât părintele său. Care este înălțimea maximă a arborelui?
9. Fie arborele cu rădăcină descris prin următorul vector $tata = (4, 8, 8, 0, 1, 4, 8, 6, 2, 6)$. Enumeră descendenții nodului **8**.

PARTEA a II-a

TEZE

Specializarea Matematică-Informatică

Teza 1

SUBIECTUL I **(20 de puncte)**

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Variabilele **a, b** și **z** sunt reale, iar **a ≤ b.** Care dintre expresiile următoare C/C++ are valoarea **1** dacă și numai dacă valoarea variabilei **x nu** aparține intervalului închis determinat de valorile variabilelor **a** și **b**?
a) `(x>a)||(x>b)` **b)** `(x<a)||(x>b)` **c)** `x<a && x>b` **d)** `x>=a && x<=b`

2. Numim înălțime a unui arbore cu rădăcină numărul de muchii ale celui mai lung lanț elementar care are una dintre extremități în rădăcina arborelui. Înălțimea arborelui cu rădăcină, având **8** noduri, numerotate de la **1** la **8**, reprezentat prin vectorul „de tați" **(6, 6, 5, 0, 6, 4, 4, 7)** este:
a) 2 **b)** 3 **c)** 4 **d)** 5

3. Se consideră algoritmul care determină toate permutările distincte de n obiecte (numerotate de la 1 la n) în care nu există puncte fixe. O permutare (p1, p2, ..., pn) are puncte fixe dacă există cel puțin o componentă pi = i. De exemplu, pentru n = 5, permutarea (2, 3, 5, 4, 1) are puncte fixe deoarece p4 = 4. Pentru n = 4, stabiliți câte permutări fără puncte fixe există.
a) 8 **b)** 12 **c)** 10 **d)** 9

4. Se consideră un graf neorientat cu 20 noduri cu proprietatea că gradul fiecărui nod este mai mare sau egal cu 3. Care este numărul maxim de componente conexe pe care le poate avea acest graf?
a) 4 **b)** 6 **c)** 1 **d)** 5

5. Se consideră subprogramul f definit de mai jos.

```
void f (int n)
  { if (n!=0)
       {if (n%10>4)cout<<n%10;
        f (n/10) ;
        cout<<n%10;
       }
  }
```

Ce se va afișa în urma apelului f(54621)?
a) 6554621 **b)** 654621 **c)** 6655421 **d)** 1266455

SUBIECTUL II (40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

```
citește a, n
(a număr întreg, n număr natural)
p←1
┌cât timp n>0 execută
│ ┌ dacă  n%2=0 atunci
│ │    a←a*a
│ │    n←[n/2]
│ altfel
│    p←p*a
│    n←n-1
└─■
└─■
scrie p
```

a) Scrieți valoarea care se va afișa pentru a = 2 și n = 7. **(6 puncte)**

b) Scrieți două seturi distincte de valori pentru a și n astfel încât rezultatul afișat să fie 1. **(6 puncte)**

c) Scrieți programul C/C++ corespunzător algoritmului dat. **(10 puncte)**

d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se utilizeze alt tip de structură repetitivă. **(6 puncte)**

2. Se consideră declarările de mai jos, în care variabila e memorează numele și data nașterii unui elev. Scrieți o secvență de instrucțiuni C/C++ care citește de la tastatură informațiile despre un elev în variabila e și afișează numele elevului dacă anul nașterii este egal cu 2000 sau data nașterii în caz contrar, informațiile din dată se afișează pe aceeași linie separate printr-un spațiu. **(6 puncte)**

```
struct data
{int zi,luna, an;};
struct elev
{ char nume[21];
  data dn;
}
elev e;
```

3. În secvența de instrucțiuni de mai jos variabilele i și j sunt de tip întreg, iar variabila A memorează un tablou bidimensional cu 5 linii și 5 coloane, numerotate de la 1 la 5. Elementele tabloului sunt numere întregi. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât în urma executării secvenței obținute, tabloul memorat în variabila A să aibă elementele din figura de mai jos. **(6 puncte)**

```
for(i=1;i<=5;i++)
  for(j=1;j<=5;j++)
     ..............
```

```
2 3 4 0 1
3 4 0 1 2
4 0 1 2 3
0 1 2 3 4
1 2 3 4 0
```


SUBIECTUL III (30 de puncte)

1. Scrieți un program C/C++ care citește de la tastatură un șir de caractere format din cel mult 100 de caractere (litere mici și mari ale alfabetului englez și caracterul #) și afișează pe ecran șirul obținut prin inversarea ordinii literelor tuturor cuvintelor de lungime maximă. Șirul începe și se termină cu caracterul #, ca în exemplu. Un cuvânt din șir reprezintă o succesiune de litere delimitate de două caractere #.
Exemplu: dacă se citește șirul de caractere **#Voi#da#bacu#la#info#** se va afișa **#Voi#da#ucab#la#ofni#** **(10 puncte)**

2. Subprogramul **numărare** are patru parametri:
- **n**, prin care primește un număr natural ($2 \le n \le 20$);
- **m**, prin care primește un număr natural ($2 \le m \le 20$);
- **a**, prin care primește un tablou unidimensional care memorează un șir de **n** numere naturale, fiecare cu cel mult 4 cifre;
- **b**, prin care primește un tablou unidimensional care memorează un șir de **m** numere naturale, fiecare cu cel mult 4 cifre.

Subprogramul returnează numărul de elemente din tabloul a, care sunt strict mai mici decât toate elementele din tabloul b. Scrieți în limbajul C/C++ definiția completă a subprogramului **numărare**.
Exemplu: dacă **n = 7**, **m = 8** și **a = (1, 4, 5, 3, 82, 6, 2)**, **b = (56, 6, 34, 23, 8, 9, 12, 18)** atunci, după apel, subprogramul va returna valoarea **5** (valorile **1, 4, 5, 3, 2** din tabloul a sunt strict mai mici decât toate valorile din tabloul b. **(10 puncte)**

3. Numim **secvență fazan** a unui șir de numere naturale un subșir al acestuia, format din termeni aflați pe poziții consecutive în șirul dat cu proprietatea că prima cifră a termenului curent este egală cu ultima cifră a termenului anterior. Lungimea secvenței este egală cu numărul de termeni ai acesteia. Fișierul **bac.txt** conține un șir de cel puțin două și cel mult 10^9 de numere naturale din intervalul $[0, 10^9]$. Numerele sunt separate prin câte un spațiu, iar în șir există cel puțin doi termeni fazan pe poziții consecutive. Se cere să se determine o **secvență fazan** de lungime maximă în șirul aflat în fișier și să se afișeze pe ecran lungimea acestei secvențe. Pentru determinarea secvenței cerute se utilizează un algoritm eficient din punctul de vedere al memoriei necesare și al timpului de executare.
Exemplu: dacă fișierul **bac.txt** conține numerele 12 **13 31 123 321** 61 76 **25 54 425 511 121** 311 311 atunci pe ecran se afișează valoarea **5.**

a) Se cere să se proiecteze un algoritm eficient din punct de vedere al timpului de executare și al spațiului de memorie utilizat și să se realizeze o descriere de 3-4 rânduri a algoritmului ales justificându-se eficiența acestuia. **(2 puncte)**

b) Scrieți programul C/C++ corespunzător algoritmului descris. **(8 puncte)**

Teza 2

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Variabilele **a, b** și **z** sunt reale. Care dintre următoarele instrucțiuni C/C++ atribuie variabilei z valoarea maximă dintre valorile variabilelor a și b?
   a) `z=a>b?a:b;`
   b) `z=a<b?a:b;`
   c) `z=(a+b)/2-(a-b)/2`
   d) `z=(a+b)/2;`

2. Într-un graf neorientat cu 10 noduri, fiecare nod are gradul 2. Care este numărul maxim de componente conexe din care poate fi format graful?
   a) 6 **b)** 3 **c)** 5 **d)** 1

3. Se generează toate submulțimile cu 2 elemente ale mulțimii {1, 2, 3, 4} în ordinea 1 2, 1 3, 1 4, 2 3, 2 4, 3 4. Dacă se utilizează exact aceeași metodă pentru a genera submulțimile de trei elemente ale mulțimii {5, 6, 7, 8}, atunci antepenultima submulțime este:
   a) 5 7 8 **b)** 5 6 8 **c)** 6 7 8 **d)** 5 6 7

4. Care este numărul maxim de arce într-un graf orientat cu 10 noduri?
   a) 100 **b)** 20 **c)** 45 **d)** 90

5. Se consideră subprogramul **f** definit astfel:

```
void f (int n, int &m)
{if (n!=0)
        {if (n%10>m)m=n%10;
        f(n/10,m) ;
        cout<<m<<' ';
        }
}
```

   Ce se va afișa pentru m = 0 în urma apelului f(54321, m)?
   a) 5 4 3 2 1 **b)** 1 2 3 4 5 **c)** 5 5 5 5 5 **d)** 5 4 3 1 2

SUBIECTUL II (40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

a) Scrieți valoarea care se va afișa dacă se citesc pe rând valorile 4, 8, 6, 16, 45. **(6 puncte)**
b) Pentru n = 4, scrieți un set de date de astfel încât rezultatul afișat să fie 0. **(6 puncte)**
c) Scrieți programul C/C++ corespunzător algoritmului dat. **(10 puncte)**
d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se utilizeze alt tip de structură repetitivă. **(6 puncte)**

```
citește n  (n număr natural)
y←0
cât timp n>0 execută
    citește x
    cât timp x%2 = 0 execută
        x←x/2
    dacă x=1 atunci
        y←y+1
    n←n-1
scrie y
```

2. Se consideră declarările de mai jos, în care variabila **p** memorează numărul de vârfuri și coordonatele carteziene ale vârfurilor unui poligon convex, variabila i este de tip int. Scrieți o secvență de instrucțiuni C/C++ care înlocuind punctele de suspensie citește de la tastatură coordonatele vârfurilor poligonului. **(6 puncte)**

```
struct punct
{int x,int y;};
struct poligon
{ int nr_vf ;
  punct vf[20];}p;

cin>>p.nr_vf;
for(i=0;i<p.nr_vf;i++)
      .....
```

3. În secvența de instrucțiuni de mai jos variabilele i și j sunt de tip întreg, iar variabila A memorează un tablou bidimensional cu 5 linii și 5 coloane, numerotate de la 1 la 5. Elementele tabloului sunt numere întregi. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, tabloul memorat în variabila A să aibă elementele din figura de mai jos. **(6 puncte)**

```
for(i=1;i<=5;i++)
  for(j=1;j<=5;j++)
     ............
```

```
1 1 1 1 3
1 1 1 3 2
1 1 3 2 2
1 3 2 2 2
3 2 2 2 2
```

SUBIECTUL III (30 de puncte)

1. Scrieți un program C/C++ care citește de la tastatură un șir de caractere format din cel mult 100 de caractere (litere mici, litere mari ale alfabetului englez și spații) și afișează pe ecran litera care apare de cele mai multe ori și numărul ei de apariții, ca în exemplu. Dacă sunt mai multe litere care apar de același număr maxim de ori se vor afișa toate.
Exemplu: Dacă se citește șirul de caractere **`Voi da Bacalaureatul la Informatica`** se va afișa **a 8**. **(10 puncte)**

2. Subprogramul **`numărare`** are doi parametri:
   - a, prin care primește un număr natural ($2 \leq a \leq 20000$);
   - b, prin care primește un număr natural ($2 \leq b \leq 20000$);

   Subprogramul returnează numărul de valori din intervalul închis determinat de a și b, care au exact 3 divizori. Scrieți în limbajul C/C++ definiția completă a subprogramului **`numărare`**.
   Exemplu: Dacă a = 6 și b = 36, subprogramul va returna 2. **(6 puncte)**

3. Din fișierul **`bac.txt`** se citesc `n` și `m` (numere naturale, `0<m<n<1000000`) de pe prima linie, apoi `n` numere naturale cu cel mult două cifre fiecare `a1, a2, ..., an` de pe linia a doua și apoi `m` numere naturale cu cel mult două cifre fiecare `b1, b2, ..., bm` de pe linia a treia a fișierului. Să se determine câte șiruri `b` se pot obține din șirul `a` dacă se poate schimba ordinea elementelor din șirul `a`. Se va afișa pe ecran numărul numărul de șiruri obținute.
   a) Se cere să se proiecteze un algoritm eficient din punct de vedere al timpului de executare și să se realizeze o descriere de 3-4 rânduri a algoritmului ales justificându-se eficiența acestuia. **(2 puncte)**
   b) Să se scrie programul C/C++ ce realizează prelucrarea descrisă și afișează pe ecran un mesaj corespunzător. **(8 puncte)**
   De exemplu, pentru fișierul **`bac.txt`** cu conținutul:
   8 3
   1 6 3 1 3 7 6 1
   6 1 3
   se afișează valoarea **2**.

Teza 3

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Variabilele **a, b** sunt numere naturale. Care dintre expresiile C/C++ are valoarea 1 dacă și numai dacă valorile variabilelor a și b sunt numere naturale consecutive?
   a) `a-b==1`
   b) `a==1&&b==0`
   c) `a-b==1&&b-a==1`
   d) `a-b==1||b-a==1`

2. Se consideră un arbore cu rădăcină, în care rădăcina se află pe nivelul 0 și orice nod de pe nivelul i are exact i + 1 descendenți, cu excepția nodurilor de pe nivelul 3 care sunt noduri terminale. Numărul de noduri frunze ale arborelui sunt:
   a) 5 **b)** 10 **c)** 6 **d)** 7

3. Se utilizează metoda backtracking pentru a genera toate submulțimile mulțimii {1, 2, 3, 4, 5}. Câte submulțimi care conțin elementul 2 și nu conțin elementul 4 sunt generate?
   a) 8 **b)** 12 **c)** 10 **d)** 16

4. Într-un graf neorientat cu 10 noduri, există muchii între nodurile `i, j` care au proprietatea că `abs(i-j)>0`. Numărul de valori egale cu 1 din matricea de adiacență este:
   a) 91 **b)** 100 **c)** 90 **d)** 80

5. Se consideră subprogramul f definit astfel:

```
void f (int n)
  { if (n!=0)
       {
            f(n/2);
            cout<<n%2;
       }
  }
```

   Ce se va afișa în urma apelului f(43)?
   a) 11011 **b)** 110101 **c)** 1101011 **d)** 101011

SUBIECTUL II (40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

```
citește x (x număr natural)
nr←0
cât timp x>0 execută
    dacă x%2=0 atunci
        nr←nr+1
    x←[x/2]
scrie nr
```

a) Scrieți valoarea care se va afișa pentru x =13 **(6 puncte)**

b) Scrieți două valori distincte x astfel încât rezultatul afișat să fie 0. **(6 puncte)**

c) Scrieți programul C/C++ corespunzător algoritmului dat. **(10 puncte)**

d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se utilizeze alt tip de structură repetitivă. **(6 puncte)**

2. Se consideră declarările de mai jos, în care variabilele **E1** și **E2** memorează numele și data nașterii a doi elevi. Scrieți o secvență de instrucțiuni C/C++ care afișează numele elevului mai mare, știind că ambii elevi sunt născuți în același an, în zile diferite. **(6 puncte)**

```
struct data
{int zi, luna, an;};
Struct elev
{ char nume[21];
data dn;}
elev E1, E2;
```

3. Ce se va afișa în urma executării secvenței de program următoare? **(6 puncte)**

```
char a[3][12]={"bacalaureat", "la", "informatica"};
cout<<a[0]<<' '<<a[1][1]<<' '<<a[2][0];
```

SUBIECTUL III (30 de puncte)

1. Scrieți un program C/C++ care citește de la tastatură un număr natural impar N (N∈[2,50]) și elementele unui tablou bidimensional cu N linii și N coloane, numere întregi, apoi transformă tabloul în memorie, ștergând o linie și o coloană, la mijlocul său ca în exemplu. Tabloul obținut se afișează pe ecran, linie cu linie, elementele fiecărei linii fiind separate prin câte un spațiu. **(10 puncte)**

Exemplu: Dacă N=5 și tabloul este:

1	2	**3**	4	5
6	7	**8**	9	4
3	**4**	**5**	**6**	**7**
3	2	**3**	4	5
6	5	**7**	8	9

atunci se obține tabloul următor:

1	2	4	5
6	7	9	4
3	2	4	5
6	5	8	9

2. Subprogramul **numar** are doi parametri:

- **n**, prin care primește un număr natural ($2 \le n \le 10^9$);
- **m**, prin care furnizează un număr natural ($2 \le m \le 10^9$) cel mai mare număr natural care se poate obține folosind toate cifrele impare care apar în scrierea lui n, dacă în scrierea lui n nu apar cifre impare atunci m va avea valoarea **–1**. Scrieți în limbajul C/C++ definiția completă a subprogramului **numar**.

Exemplu: Dacă **n=74317** atunci **m=7731**, dacă **n=246** atunci **m=-1**. **(10 puncte)**

3. Scrieți un program C/C++ care citește de la tastatură trei numere naturale n, m și k. Programul afișează în fișierul **bac.txt** numerele naturale cu exact k cifre care sunt divizibile atât cu n cât și cu m ($1 \le n, m \le 10^9$, $1 \le k \le 9$).

Exemplu: Dacă n = 15, m = 10, k = 2, atunci în fișier se vor afișa numerele 30 60 90.

a) Se cere să se proiecteze un algoritm eficient din punct de vedere al timpului de executare și al spațiului de memorie utilizat și să se realizeze o descriere de 3-4 rânduri a algoritmului ales justificându-se eficiența acestuia. **(2 puncte)**

b) Scrieți programul C/C++ corespunzător metodei descrise la **a)**. **(8 puncte)**

Teza 4

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Valoarea variabilei **a** este un umăr natural. Care dintre expresiile C/C++ următoare are valoarea **1** dacă și numai dacă valoarea variabilei **a are exact 3 cifre.**
   a) a/1000==0
   b) a/100!=0
   c) a/1000==0&&a/100!=0
   d) a/100==0&&a/10!=0

2. Un arbore cu rădăcină este reprezentat prin următorul vector de „tați" **(6, 6, 5, 0, 6, 4, 4, 2),** determinați nodurile care pot fi alese ca rădăcină astfel încât nodul 6 să aibă un număr maxim de descendenți.
   a) 1, 2, 5 **b)** 3, 8 **c)** 4,7 **d)** 6

3. Se generează în ordine lexicografică toate șirurile de lungime 3 cu structura vocală – consoană – vocală formate din literele **a, b, c, d, e**. Știind că primele 3 șiruri generate sunt **aba, abe, aca,** care dintre următoarele secvențe reprezintă o secvență de șiruri generate unul după altul?
   a) eba, ebe, ece
   b) eca, ebe, ece
   c) eca, ece, ede
   d) ebe, eca, ece

4. Care este numărul de grafuri neorientate cu 5 vârfuri cu proprietatea că vârfurile 1 și 2, respectiv 1 și 3 nu sunt adiacente?
   a) 32 **b)** 4^3 **c)** 2^8 **d)** 2^6

5. Se consideră subprogramul **f** definit de mai jos.

```
int f (int n)
{
     if (n%10==5) return 2;
     else return 2*f(n+1);
}
```

Ce se va afișa în urma apelului **f(1)**?
a) 16 **b)** 32 **c)** 64 **d)** 8

SUBIECTUL II (40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

```
citește x
(x număr întreg)
y←0
pentru i←9,0,–1 execută
  cx←x
  cât timp cx>0 execută
    dacă cx%10=i atunci
      y← y*10+i
    cx←cx/10
scrie y
```

a) Scrieți valoarea care se va afișa pentru x= 7172. **(6 puncte)**
b) Scrieți două seturi distincte de valori pentru x astfel încât rezultatul afișat să fie 4332. **(6 puncte)**
c) Scrieți programul C/C++ corespunzător algoritmului dat. **(10 puncte)**
d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască structura pentru cu alt tip de structură repetitivă. **(6 puncte)**

2. Se consideră declarările de mai jos, în care variabilele z memorează partea reală și partea imaginară a unui număr complex. Scrieți o secvență de instrucțiuni C/C++ care citește de la tastatură informațiile despre numărul complex z și afișează pe ecran modulul numărului complex memorat în variabila z. **(6 puncte)**

```
struct complex
{ int pre;
int pim;}
complex z;
```

3. În secvența de instrucțiuni de mai jos variabilele i și j sunt de tip întreg, iar variabila A memorează un tablou bidimensional cu 5 linii și 5 coloane, numerotate de la 1 la 5. Elementele tabloului sunt numere întregi. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, tabloul memorat în variabila A să aibă elementele din figura de mai jos. **(6 puncte)**

```
for(i=1;i<=5;i++)
  for(j=1;j<=5;j++)
    ............
```

```
1 1 1 1 1
1 2 3 4 5
1 3 6 0 5
1 4 0 0 5
1 5 5 5 0
```

SUBIECTUL III (30 de puncte)

1. Scrieți un program C/C++ care citește de la tastatură un șir de caractere format din cel mult 100 de caractere (litere mici și mari ale alfabetului englez și caracterul spațiu) și afișează pe ecran șirul obținut prin inserarea unui număr minim de caractere #, astfel încât să nu existe în șir două caractere alăturate identice.
   Exemplu: Dacă se citește șirul `copiii au acces` se va afișa `copi#i#i au ac#ces`
   (10 puncte)

2. Subprogramul **numărare** are doi parametri:
   - **n**, prin care primește un număr natural ($1 \le n \le 10^9$);
   - **m**, prin care primește un număr natural ($1 \le m \le 10^9$, $n \le m$).

   Subprogramul returnează numărul de cifre utilizate pentru construirea numerelor naturale din intervalul `[n, m]`. Scrieți în limbajul C/C++ definiția completă a subprogramului **numărare**.
   Exemplu: Dacă **n = 7** și **m = 56** atunci, după apel, subprogramul va returna valoarea **115.**
   (10 puncte)

3. Scrieți un program C/C++ care citește din fișierul text **BAC.TXT** un șir cu cel mult 100 000 de numere întregi formate din cel mult 2 cifre fiecare și afișează pe ecran separate printr-un spațiu, numărul sau numerele din fișier cu număr maxim de apariții.
   Exemplu: Dacă în fișier sunt numerele `27 -8 43 27 -8 9 10 43` atunci pe ecran se vor afișa nu neapărat în această ordine `-8 43 27.`
   a) Se cere să se proiecteze un algoritm eficient din punct de vedere al timpului de executare și al spațiului de memorie utilizat și să se realizeze o descriere de 3-4 rânduri a algoritmului ales justificându-se eficiența acestuia. **(2 puncte)**
   b) Scrieți programul C/C++ corespunzător metodei descrise la punctul **a)**. **(8 puncte)**

Teza 5

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Variabilele **a, b** și **z** sunt reale, iar **a≤b.** Care dintre expresiile C/C++ următoare are valoarea **1** dacă și numai dacă valoarea variabilei **z aparține** intervalului închis determinat de valorile variabilelor **a** și **b**?
   a) `(z>a||(z>b)`
   b) `(z<a)||(z>b)`
   c) `!(z<a && z>b)`
   d) `!(z<a || z>b)`

2. Un arbore cu rădăcină are exact 2019 noduri. Știind că vectorul de tați T al acestui arbore are proprietatea că T[i]=[i/2], pentru oricare i de la 1 la 2019, atunci numărul de noduri care au exact un descendent este:
   a) 2 **b)** 2018 **c)** 1 **d)** 0

3. Se consideră algoritmul care determină toate parfumurile formate din 3 esențe din mulțimea {mosc, santal, tuberoze, liliac, lavandă}. Două parfumuri sunt diferite dacă diferă prin cel puțin o esență. Care este numărul de parfumuri care se pot realiza?
   a) 8 **b)** 12 **c)** 10 **d)** 9

4. Se consideră un graf orientat cu 7 noduri și 2 componente tare conexe, o componentă cu 4 noduri și o componentă cu 3 noduri. Numărul maxim de arce care pot exista în graf este:
   a) 14 **b)** 25 **c)** 30 **d)** 20

5. Se consideră subprogramul **f** definit mai jos.

```
int f (int n)
    {
        if (n==0)return 0;
        else if(n%3==0)return 1+f(n/10);
             else return f(n/10);
    }
```

   Ce se va afișa în urma apelului f(23160)?
   a) 2 **b)** 1 **c)** 3 **d)** 4

SUBIECTUL II (40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

```
citește x
(x număr natural)
┌cât timp x>9 execută
│ y←0
│ ┌ cât timp x>0 execută
│ │    y←y+x%10
│ │    x←[x/10]
│ └■
│ x←y
└■
scrie x
```

a) Scrieți valoarea care se va afișa pentru x = 2019. **(6 puncte)**

b) Scrieți cel mai mic număr de două cifre și cel mai mare număr de două cifre care pot fi citite pentru x astfel încât rezultatul afișat să fie 9. **(6 puncte)**

c) Scrieți programul C/C++ corespunzător algoritmului dat. **(10 puncte)**

d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să nu se utilizeze structuri repetitive. **(6 puncte)**

2. Se consideră declarările de mai jos, în care variabila p memorează simultan numărul de vârfuri ale unui poligon (număr natural din intervalul $[3,10^2]$) și coordonatele vârfurilor acestuia (abscisa și ordonata) în sistemul de coordonate xOy (numere reale). Fără a utiliza variabile suplimentare scrieți o secvență de instrucțiuni C/C++ care afișează pe ecran perimetrul poligonului memorat în variabila p. **(6 puncte)**

```
struct punct
{float x,y;};
struct poligon
{ int nr;
punct v[101];}p;
int i;
float per;
```

3. În secvența de instrucțiuni de mai jos variabilele **i** și **j** sunt de tip întreg, iar variabila **a** memorează un tablou bidimensional cu 5 linii și 5 coloane, numerotate de la 1 la 5. Elementele tabloului sunt numere întregi. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, tabloul memorat în variabila **a** să aibă elementele din figura de mai jos. **(6 puncte)**

```
for(i=1;i<=5;i++)
  for(j=1;j<=5;j++)
    ..............
```

```
0 1 1 1 0
4 0 1 0 2
4 4 0 2 2
4 0 3 0 2
0 3 3 3 0
```


SUBIECTUL III (30 de puncte)

1. Scrieți un program C/C++ care citește de la tastatură un șir de caractere format din cel mult 100 de caractere (litere mici și mari ale alfabetului englez) și afișează pe ecran pe aceeași linie separate prin spații șirurile obținute prin concatenarea prefixelor și sufixelor de aceeași lungime.
Exemplu: Dacă se citește șirul de caractere **info** atunci pe ecran se va afișa **io info infnfo infoinfo.** **(10 puncte)**

2. Subprogramul **divizori** are un parametru **n**, prin care primește un număr natural ($2 \le n \le 10^4$). Subprogramul returnează suma divizorilor primi ai lui **n**. Scrieți în limbajul C/C++ definiția completă a subprogramului **divizori**.
Exemplu: Dacă **n = 12** atunci, după apel, subprogramul va returna valoarea **5**, divizorii primi ai lui 12 sunt 2 și 3. **(10 puncte)**

3. Scrieți un program C/C++ care citește din fișierul text **BAC.TXT** un șir S cu cel mult 100 000 de numere naturale formate din cel mult trei cifre fiecare. Asupra acestui șir se aplică în mod repetat următoarea prelucrare: se elimină din șir valorile prime, iar valorile neprime se incrementează cu valoarea 1. Prelucrarea se repetă până când în șir nu mai există niciun număr. Să se afișeze pe ecran de câte ori a fost aplicată această prelucrare.
Exemplu: Dacă fișierul conține numerele 12 **11** 16 45 34 atunci după prima prelucrare vom avea valorile **13 17** 46 35; după a doua **47** 36; după a treia **37**; pe ecran va fi afișată valoarea **4.**

a) Se cere să se proiecteze un algoritm eficient din punct de vedere al timpului de execuare și al spațiului de memorie utilizat și să se realizeze o descriere de 3-4 rânduri a algoritmului ales justificându-se eficiența acestuia. **(2 puncte)**

b) Scrieți programul C/C++ corespunzător metodei descrise la punctul **a)**. **(8 puncte)**

Teza 6

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Variabilele **a** și **b** memorează numere naturale. Care dintre expresiile C/C++ următoare are valoarea **1** dacă și numai dacă valorile variabilelor **a** și **b** au aceeași paritate?
   1. `(a+b)%2==0`
   2. `(a-b)%2==0`
   3. `a%2==b%2`
   4. `a%2+b%2==0`

   a) 1, 2 **b)** 1, 2, 3, **c)** 2, 3, 4 **d)** 3, 4

2. Un graf neorientat cu 2019 vârfuri G = (X, U), unde X = {1, 2, …, 2019}, are proprietatea că două vârfuri i, j din graf sunt adiacente dacă și numai dacă `i<j` și `j=2*i` sau `j=2*i+1`. Câte muchii are graful?
   a) 1009 **b)** 2018 **c)** 2019 **d)** 2020

3. Într-o urnă sunt 5 bile albe și 3 bile negre. Care este numărul configurațiilor distincte care se pot obține prin extragerea bilelor din urnă și așezarea lor în linie în ordinea extragerii.
   a) 54 **b)** 28 **c)** 120 **d)** 56

4. Se consideră un graf neorientat cu 12 noduri și 3 componenete conexe. Numărul maxim de muchii care pot exista în graf este:
   a) 15 **b)** 20 **c)** 55 **d)** 45

5. Se consideră subprogramul f definit mai jos.

```
void f (int n)
{
    if(n)
    {
        for(int i=1; i<=n/2; i++)
            if(n%i==0)cout<<i;
        f(n-2);
    }
}
```

   Ce se va afișa în urma apelului f(8)?
   a) 12412312 **b)** 124123121 **c)** 1234121 **d)** 1241212

SUBIECTUL II (40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

```
citește x
(x număr întreg)
y←0
pentru i←2, x/2 execută
    dacă x% i=0 atunci
        y←y+1
dacă y=0 atunci
    scrie 1
altfel
    scrie y
```

   a) Scrieți valoarea care se va afișa pentru x = 45 **(6 puncte)**
   b) Scrieți toate numerele din intervalul [1, 9] care pot fi citite pentru x astfel încât să se afișeze valoarea 1. **(6 puncte)**
   c) Scrieți programul C/C++ corespunzător algoritmului dat. **(10 puncte)**
   d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască structura pentru cu alt tip de structură repetitivă. **(6 puncte)**

2. Se consideră declarările de mai jos, în care variabilele A și B sunt coordonatele a două puncte (abscisa și ordonata) în sistemul de coordonate carteziene xOy (numere reale). Fără a utiliza variabile suplimentare scrieți o expresie C/C++ care are valoarea 1 dacă și numai dacă segmentul descris de punctele A și B se află pe prima bisectoare a sistemului de coordonate carteziene xOy. **(6 puncte)**

```
struct punct
{float x,y;}A, B;
```

3. În secvența de instrucțiuni de mai jos variabilele i și j sunt de tip întreg, iar variabila A memorează un tablou bidimensional cu 5 linii și 5 coloane, numerotate de la 1 la 5. Elementele tabloului sunt numere întregi. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, tabloul memorat în variabila A să aibă elementele din figura de mai jos. **(6 puncte)**

```
for(i=1;i<=5;i++)
  for(j=1;j<=5;j++)
    ..............
```

```
1  6  11 16 21
2  7  12 17 22
3  8  13 18 23
4  9  14 19 24
5  10 15 20 25
```

SUBIECTUL III (30 de puncte)

1. Scrieți un program C/C++ care citește de la tastatură un șir de caractere format din cel mult 100 de caractere (litere mici și spații) și afișează pe ecran lungimea maximă a unui cuvânt și numărul cuvintelor de lungime maximă din șir.
   Exemplu: eu dau bacalaureat la informatica atunci pe ecran se va afișa **11 2.** **(10 puncte)**

2. Subprogramul **divizor** are doi parametri un parametru **n**, prin care primește un număr natural (**2≤n≤100**) și un parametru **a**, prin care primește un tablou unidimensional care memorează un șir de **n** numere naturale, fiecare cu cel mult **4** cifre. Subprogramul returnează cel mai mare divizor comun al celor n numere naturale din tablou. Scrieți în limbajul C/C++ definiția completă a subprogramului **divizor**.
Exemplu: Dacă **n = 4** și tabloul memorează valorile **60 45 30 10** atunci, după apel, subprogramul va returna valoarea **5**. **(10 puncte)**

3. Un număr natural N este **p-compus**, dacă se poate scrie ca sumă de p numere naturale **consecutive**. Scrieți un program C/C++ care citește din fișierul text **BAC.TXT** de pe prima linie un număr natural p și de pe următoarele linii un șir S cu cel mult 100 000 de numere naturale cu cel mult 9 cifre fiecare, să se afișeze pe ecran pe aceeași linie separate printr-un spațiu, primul număr din sumă, dacă numărul din șir este p-compus sau mesajul NU în caz contrar.
Exemplu: Dacă fișierul conține numerele:
3
21 19 16 12 atunci pe ecran se va afișa **6 NU NU 3.**
a) Se cere să se proiecteze un algoritm eficient din punct de vedere al timpului de executare și al spațiului de memorie utilizat și să se realizeze o descriere de 3-4 rânduri a algoritmului ales justificându-se eficiența acestuia. **(2 puncte)**
b) Scrieți programul C/C++ corespunzător metodei descrise la **a)**. **(8 puncte)**

Teza 7

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Variabilele a, b, n memorează numere naturale. Care dintre expresiile C/C++ următoare are valoarea **1** dacă și numai dacă rădăcina pătrată a lui n nu depășește valoarea variabilei a și nici valoarea variabilei b?
a) `sqrt(n)<=a&&sqrt(n)>=b`
b) `sqrt(n)<=a&&sqrt(n)<=b`
c) `sqrt(n)<=a||sqrt(n)<=b`
d) `sqr(n)<=a&&sqr(n)<=b`

2. Un arbore cu n vârfuri este reprezentat prin matricea de adiacență. Numărul de elemente egale cu 1 din matricea de adiacență asociată grafului este:
a) n **b)** n^2-n **c)** $n^2-2*n+2$ **d)** $2*n-2$

3. Un copil are în biblioteca personală 10 cărți în limba română, 4 cărți în limba germană și 6 cărți în limba engleză. Acesta trebuie să ducă la școală două cărți scrise în limbi diferite. Câte posibilități de alegere a cărților are?
a) 120 **b)** 124 **c)** 240 **d)** 128

4. Fiind dat un graf orientat cu 10 noduri și fără circuite atunci numărul maxim de arce pe care le poate avea graful este:
a) 110 **b)** 55 **c)** 45 **d)** 90

5. Se consideră subprogramul f definit mai jos.

```
void f (int n)
  {
      if(n)
       {
           f(n/3);
           cout<<n%3;
       }
  }
```

Ce instrucțiune de apel trebuie executată astfel încât în urma apelului să se afișeze 1012?
a) f(29) **b)** f(34) **c)** f(30) **d)** f(32)

SUBIECTUL II (40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

```
citește x, y
(x, y numere naturale)
r←x
┌cât timp y ≤ r execută
│ r←r-y
└■
k←(x-r)/y
scrie k, r
```

a) Scrieți valoarea care se va afișa pentru x = 140 și y = 15. **(6 puncte)**

b) Scrieți două valori care pot fi citite pentru x și y astfel încât valoarea variabilei k să fie 9. **(6 puncte)**

c) Scrieți programul C/C++ corespunzător algoritmului dat. **(10 puncte)**

d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să nu se utilizeze nicio structură repetitivă. **(6 puncte)**

2. Se consideră declarările de mai jos, în care variabilele A și B sunt coordonatele a două puncte (abscisa și ordonata) în sistemul de coordonate carteziene xOy (numere reale). Fără a utiliza variabile suplimentare scrieți o expresie C/C++ care are valoarea 1 dacă și numai dacă segmentul descris de punctele A și B se află pe una dintre axele sistemului de coordonate carteziene xOy. **(6 puncte)**

```
struct punct
{float x,y;}A, B;
```

3. În secvența de instrucțiuni de mai jos variabilele i și j sunt de tip întreg, iar variabila A memorează un tablou bidimensional cu 5 linii și 5 coloane, numerotate de la 1 la 5. Elementele tabloului sunt numere întregi. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, tabloul memorat în variabila A să aibă elementele din figura de mai jos. **(6 puncte)**

```
for(i=1;i<=5;i++)
  for(j=1;j<=5;j++)
    ..............
```

```
1  2  3  4  5
10 9  8  7  6
11 12 13 14 15
16 17 18 19 20
25 24 23 22 21
```

SUBIECTUL III (30 de puncte)

1. Scrieți un program C/C++ care citește de la tastatură două șiruri de caractere S și C de aceeași lungime, formate din cel mult 100 de caractere, șirul S este format din litere mici ale alfabetului englez, iar șirul C este format din cifre, programul codifică șirul S, înlocuind fiecare literă S[i] din șir cu litera din alfabet aflată la distanța C[i], dacă șirul literelor se termină, se reia de la litera a. Șirul astfel obținut se afișează pe ecran.
Exemplu: Dacă se citesc șirurile **info 3212** atunci se va afișa șirul **lpgr.** **(10 puncte)**

2. Subprogramul **permuta** are trei parametri: un parametru n, prin care primește un număr natural (2≤n≤100), un parametru a, prin care primește un tablou unidimensional care memorează un șir de n numere naturale, fiecare cu cel mult 4 cifre și un parametru k prin care primește un număr natural (1≤k<n). Subprogramul **permută circular la stânga** cu **k** poziții cele **n** numere naturale din tablou. Scrieți în limbajul C/C++ definiția completă a subprogramului **permuta**.
Exemplu: Dacă n = 4, k = 2 și tabloul memorează valorile 60 45 30 10 atunci, după apel tabloul va memora **30 10 60 45**. **(10 puncte)**

3. Scrieți un program C/C++ care citește din fișierul text **BAC.IN** un șir S cu cel mult 100 000 de numere naturale de forma 10^k, unde **k** este un număr natural din intervalul [0,9], programul afișează în fișierul **BAC.OUT** elementele șirului S în ordine crescătoare.
Exemplu: Dacă fișierul **BAC.IN** conține numerele 100 10 100 1 10 atunci în fișierul **BAC.OUT** vor fi afișate valorile **1 10 10 100 100.**

a) Se cere să se proiecteze un algoritm eficient din punct de vedere al timpului de executare și al spațiului de memorie utilizat și să se realizeze o descriere de 3-4 rânduri a algoritmului ales justificându-se eficiența acestuia. **(2 puncte)**

b) Scrieți programul C/C++ corespunzător metodei descrise la punctul **a)**. **(8 puncte)**

Teza 8

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Variabilele n, k, nr memorează numere naturale. Variabila `nr` trebuie să memoreze numărul de numere din intervalul `[0,n]` care sunt divizibile cu `k`. Cu ce expresie trebuie completată atribuirea `nr=...`?
a) `nr=n/k`
b) `nr=n/k+1`
c) `nr=n/k+n%k`
d) `nr=n%k+1`

2. Se dă graful neorientat complet cu n noduri, câte noduri va avea un arbore cu același număr de muchii ca graful dat?
a) n*(n + 1)/2
b) $(n^2 - n + 2)/2$
c) n*(n + 1)/2 + 1
d) n*(n – 1)/2 – 1

3. La examenul de Bacalaureat la Subiectul I sunt cinci itemi de tip grilă, fiecare item având 4 variante de răspuns. În câte moduri pot fi completate răspunsurile la acest subiect, dacă există posibilitatea de a nu completa niciun răspuns?
a) 5^5 **b)** 4^5 **c)** 5^2 **d)** 4^2

4. Se consideră un graf neorientat cu 10 noduri și 2 componenete conexe, o componentă cu 7 noduri și o componentă cu 3 noduri. Numărul minim de muchii care pot exista în graf este:
a) 10 **b)** 8 **c)** 16 **d)** 14

5. Se consideră subprogramul f definit mai jos.

```
void f (int n)
{
        cout<<n;
        for(int i=1; i<=n/2; i++)
             if(n%i==0)f(i);
}
```

Ce se va afișa în urma apelului f(6)?
a) 6123121 **b)** 612312 **c)** 612131 **d)** 61213

SUBIECTUL II (40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

```
citește x
(x număr natural, x≠1)
y←0
pentru i←1, x/2 execută
    dacă x% i=0 atunci
        y←y+i
dacă y=x atunci
    scrie 1 ,' ', x
altfel
    scrie y
```

a) Scrieți valoarea care se va afișa pentru x = 12. **(6 puncte)**
b) Scrieți o valoare care poate fi citită pentru x astfel încât rezultatul afișat să fie 1,' ', x. **(6 puncte)**
c) Scrieți programul C/C++ corespunzător algoritmului dat. **(10 puncte)**
d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască structura **pentru** cu alt tip de structură repetitivă cu test final. **(6 puncte)**

2. Se consideră declarările de mai jos, în care variabila t memorează simultan lungimile laturilor unui triunghi (numere reale pozitive). Fără a utiliza variabile suplimentare, scrieți o expresie C/C++ care are valoarea 1 dacă și numai dacă trunghiul ale cărui laturi sunt memorate în variabila t este un triunghi isoscel, fără a fi triunghi echilateral. **(6 puncte)**

```
struct triunghi
{float a, b, c;} t;
```

3. În secvența următoare variabilele **n** și **i** sunt de tip întreg, iar variabila **s** permite memorarea unui cuvânt, șir de cel mult 10 de caractere. Cuvintele citite sunt formate din cifre zecimale și reprezintă numere de telefon, acestea sunt separate prin Enter.

```
n=.....;
for(i=1;i<=5;i++)
{
 cin>>s;
 ............
}
```

Fără a utiliza alte variabile, scrieți secvența de instrucțiuni care poate înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, variabila **n** să memoreze numărul de șiruri citite pentru care subșirul format din primele patru caractere ale lor coincide cu șirul 2019, iar acesta NU mai apare pe alte poziții în codul respectiv. **(6 puncte)**
Exemplu: Dacă se citesc șirurile, variabila **n** are valoarea 2.
2019745432
1378912217
2019345435
2019782019
1765412019

SUBIECTUL III (30 de puncte)

1. Scrieți un program C/C++ care citește de la tastatură un număr natural N (N∈[2,10]) și elementele unui tablou bidimensional cu N linii și N coloane, numere naturale din mulțimea{0, 1}, programul determină pentru fiecare linie din tablou numărul natural scris în baza 10 a cărui scriere în baza 2 pe N poziții binare este reprezentată de linia respectivă și afișează pe ecran cu spații între ele numerele determinate. **(10 puncte)**

   Exemplu: Dacă N=4 și tabloul este:
   ```
   0 1 0 1
   1 0 0 0
   1 0 0 1
   1 1 0 0
   ```
   atunci pe ecran se vor afișa valorile `5 8 9 12`.

2. Subprogramul **patrat** are un parametru **n**, prin care primește un număr natural ($2 \le n \le 10^9$). Subprogramul returnează cel mai mare pătrat perfect care este divizor al lui **n**. Scrieți în limbajul C/C++ definiția completă a subprogramului **patrat**.
   Exemplu: Dacă **n** = 72 atunci, după apel, subprogramul va returna valoarea **36**.
   (10 puncte)

3. Scrieți un program C/C++ care citește din fișierul text **BAC.TXT** un șir S cu cel mult 100 000 de numere naturale din intervalul **[2, 10^9]**. Pentru fiecare valoare x din șir se determină numărul de cifre egale cu zero de la sfârșitul lui x! (**unde x!=1*2*3...*x**, programul determină și afișează ecran cu spațiu între ele numărul de valori x din șir care au un număr maxim de valori egale cu zero la sfârșitul lui **x!** și care este această valoare maximă.
   Exemplu: Dacă fișierul conține numerele `102 12 50 100` atunci pe ecran vor fi afișate valorile `2 24`.
   a) Se cere să se proiecteze un algoritm eficient din punct de vedere al timpului de executare și al spațiului de memorie utilizat și să se realizeze o descriere de 3-4 rânduri a algoritmului ales justificându-se eficiența acestuia. **(2 puncte)**
   b) Scrieți programul C/C++ corespunzător metodei descrise la punctul **a)**. **(8 puncte)**

Teza 9

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Variabilele **x, y și z** au valori numere naturale de cel mult 4 cifre. Care dintre expresiile C/C++ următoare are valoarea **1** dacă și numai dacă exact una dintre valori este impară?
   a) `x%2 || y%2 || z%2`
   b) `(x+y)%2==0 || (x+z)%2==0 || (y+z)%2==0`
   c) `(x%10 + y%10 + z%10)%2>0`
   d) `x%10%2 + y%10%2 + z%10%2==1`

2. Numim înălțime a unui arbore cu rădăcină numărul de muchii ale celui mai lung lanț elementar care are una dintre extremități în rădăcina arborelui. Într-un arbore cu 9 noduri, toate frunzele sunt pe ultimul nivel și fiecare nod care nu este frunză, are cel mult trei fii. Care dintre următoarele variante poate fi vectorul de tați al unui astfel de arbore cu înălțimea minimă?
   T1 = (1, 2, 0, 3, 3, 2, 2, 3, 1),
   T2 = (0, 1, 2, 1, 2, 1, 4, 6, 4),
   T3 = (8, 8, 8, 1, 2, 2, 3, 9, 0),
   T4 = (6, 7, 6, 7, 0, 5, 5, 6, 7)

   a) T1, T2, T4 **b)** T2, T3 **c)** T2, T4 **d)** T2, T3, T4

3. Utilizând metoda backtracking, se generează în ordine lexicografică, toate modalitățile de a repartiza 5 studenți, în două amfiteatre A și B, astfel încât să fie cel mult 3 studenți într-un amfiteatru. Studenții sunt numerotați crescător de la 1 la 5, iar ordinea lor într-un amfiteatru nu contează. Primele 4 variante, în ordinea în care au fost generate, sunt: AAABB, AABAB, AABBA, AABBB. Care sunt cele două soluții ce vor fi generate imediat înainte și imediat după soluția BABAB?
   a) BABAA, BABBB
   b) BABAA, BABBA
   c) BAABB, BABBA
   d) BAABA, BBAAA

4. Matricea de adiacență a unui graf neorientat cu 20 de noduri, fără vârfuri izolate, conține 24 de valori egale cu 1. Care este numărul minim de componente conexe ale grafului?
   a) 8 **b)** 6 **c)** 10 **d)** 1

5. Se consideră subprogramul **f** definit mai jos.
   ```
   int f(int x) {
       int n;
       if (x>0)
   ```

```
        { if(x%2==0) { cout<<x%10; n=1+f(x/10);}
          else { n=1+f(x/10); cout<<x%10;}
          return n;}
      else return 0;
}
```

Ce se va afișa în urma apelului `cout<<f(8172)`?

a) 2817 **b)** 27184 **c)** 28174 **d)** 8217

SUBIECTUL II (40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

a) Scrieți valoarea care se va afișa pentru n = 39. **(6 puncte)**

b) Scrieți toate valorile de o cifră ce pot fi date pentru n astfel încât rezultatul afișat să fie 1. **(6 puncte)**

c) Scrieți programul C/C++ corespunzător algoritmului dat. **(10 puncte)**

d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se utilizeze cel mult o structură repetitivă. **(6 puncte)**

```
citește n (n număr natural nenul)
a←1
b←[n / 2]
┌cât timp a ≠ 0 și b>0 execută
│  c←n
│ ┌cât timp c>=b execută
│ │   c←c – b
│ └■
│  a←c
│  b←b – 1
└■
b←b+1
scrie b
```

2. Se consideră declarările C/C++ de mai jos.

```
struct produs  {
    int cod;
    float  cant,pret;
};
struct magazin {
    produs P[100];
    int nrProduse;
} M;
```

unde variabila M memorează produsele dintr-un magazin și nrProduse disponibile (maxim 100 produse). Scrieți o secvență de instrucțiuni C/C++ care afișează pentru magazinul M, pentru fiecare dintre cele nrProduse, pe câte o linie, valoarea totală a stocului (cantitate * preț), dacă acesta are cantitatea diferită de 0, respectiv codul produsului, în caz contrar. **(6 puncte)**

3. În secvența de instrucțiuni de mai jos variabilele **i** și **j** sunt de tip întreg, iar variabila A memorează un tablou bidimensional cu 6 linii și 6 coloane, numerotate de la 1 la 5. Elementele tabloului sunt numere întregi. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, tabloul memorat în variabila A să aibă elementele din figura de mai jos. **(6 puncte)**

```
for(i=0;i<=5;i++)
  for(j=5;j>=0;j--)
      ..............
```

```
6 5 5 5 5 5
5 6 3 2 1 0
3 3 6 3 3 3
5 4 3 6 1 0
1 1 1 1 6 1
5 4 3 2 1 6
```

SUBIECTUL III (30 de puncte)

1. Scrieți un subprogram C/C++, denumit **circular**, care primește prin parametrii a și b două numere naturale, nenule, de cel mult 9 cifre. Subprogramul va returna cel mai mic număr de permutări circulare la dreapta, cu câte o poziție, ale cifrelor lui a, astfel încât să se obțină numărul b. Dacă nu e posibil, funcția va returna –1.
Exemplu: Pentru a = 120362 și b = 621203 subprogramul va returna 2, iar pentru a = 732 și b = 237 va returna –1. **(10 puncte)**

2. Spunem că două cuvinte A și B **rimează**, dacă sufixul lui A care începe cu ultima vocală coincide cu sufixul lui B care începe cu ultima vocală. De exemplu, cuvintele **savant** și **gigant** rimează, în timp ce sport și cert nu rimează. Scrieți un program C/C++ care citește de la tastatură, de pe prima linie S, un text de lungime maximă 255 caractere, litere mici și spații, iar de pe a doua linie, un cuvânt C, cu cel puțin 3 și cel mult 30 de litere, dintre care cel puțin una este vocală. Textul este format din cuvinte separate prin unul sau mai multe spații. Programul va afișa, pe linii diferite, cuvintele din text care rimează cu C sau mesajul NU EXISTA, dacă S nu conține astfel de cuvinte. **(10 puncte)**

De exemplu, dacă se citesc șirurile de mai jos

```
true is bool and school is cool
tool
```

se va afișa:

```
bool
school
cool
```

3. Un număr este **palindrom** dacă citind cifrele numărului de la dreapta la stânga și de la stânga la dreapta se obține același număr. De exemplu 12321, 2002 sunt palindroame, iar 12312 nu este palindrom.
Fișierul **text bac.txt** conține pe prima linie n, un număr natural nenul, mai mic decât 105, iar pe a doua linie un șir de n numere din intervalul [1, 9], separate prin câte un spațiu. Se cere afișarea pe ecran a celui mai mare palindrom par care se poate forma cu toate cifrele date. Dacă nu există, se va afișa –1. Proiectați un algoritm eficient din punct de vedere al memoriei utilizate și al timpului de executare.
Exemplu: Dacă fișierul **bac.txt** conține numerele
`10`
`2 3 3 8 9 2 3 9 8 3` atunci pe ecran se afișează: `8933223398`.
a) Descrieți în limbaj natural algoritmul utilizat, justificând eficiența acestuia. **(2 puncte)**
b) Scrieți programul C/C++ corespunzător algoritmului descris la punctul **a)**. **(8 puncte)**

Teza 10

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Variabilele **a, b** au valori numere naturale nenule, **a ≤ b.** Care dintre următoarele instrucțiuni C/C++ atribuie variabilei întregi **c** numărul de valori pare din intervalul [a, b]?
a) `c=(b-a+1)/2;`
b) `c=a/2-b/2;`
c) `c=b/2-(a-1)/2`
d) `c=(a+b)/2;`

2. Într-un graf orientat cu 11 noduri, toate arcele sunt de forma (x,y), unde x și y au aceeași paritate și x<y. Care este numărul minim de arce de adăugat astfel încât graful să devină tare conex?
a) 4 **b)** 2 **c)** 1 **d)** 0

3. Utilizând metoda backtracking, se generează toate cadourile formate din câte 3 obiecte distincte din mulțimea **{carte, tabletă, joc, stilou, ceas, patine}**, astfel încât să nu se găsească în același cadou **tabletă** și **joc** sau **carte** și **stilou.** Primele patru soluții obținute sunt, în această ordine: **(carte, tabletă, ceas) (carte, tabletă, patine), (carte, joc, stilou)** și **(carte, joc, patine).** Indicați a șasea soluție generată.
a) `tabletă, joc, patine` **b)** `joc, stilou, ceas`
c) `carte, stilou, ceas` **d)** `tabletă, stilou, ceas`

4. Se consideră următorul arbore dat prin vectorul de tați: T = (8, 9, 9, 9, 4, 4, 5, 5, 0, 5). Câte noduri pot fi alese ca rădăcină astfel încât înălțimea arborelui să fie maximă?
a) 3 **b)** 5 **c)** 2 **d)** 6

5. Se consideră subprogramul f definită astfel:
```
void scrie(int a, int b){
  cout<<b;
  if(b){ scrie(b,a%b);
         cout<<a/b;}}
```
Ce se va afișa la apelul `scrie(8,12)`?
a) `12840210` **b)** `128421` **c)** `1248210` **d)** `1284210`

SUBIECTUL II (40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y.

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

a) Scrieți valoarea care se va afișa dacă se citesc pe rând valorile 4, 8, 16, 10, 24. **(6 puncte)**

b) Pentru n = 3, dați exemplu de trei numere naturale, nenule și distincte ce pot fi citite în variabila x, pentru care algoritmul să scrie 1. **(6 puncte)**

c) Scrieți programul C/C++ corespunzător algoritmului dat. **(10 puncte)**

d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să înlocuiți structura „repetă până când" cu o structură repetitivă condiționată anterior. **(6 puncte)**

2. Se consideră declarările C/C++ de mai jos, în care variabila **E** memorează informațiile despre jucătorii unei echipe iar variabila **i** este de tip int. Scrieți o secvență de instrucțiuni C/C++ care, înlocuind punctele de suspensie, citește de la tastatură numărul de jucători ai echipei E și informațiile despre aceștia. **(6 puncte)**

```
struct Jucator {
  int nrTricou, marimeTricou;
};
struct Echipa
{ int nrJucatori ;
  Jucator juc[20];}E;

cin>>.....;
for(i=0;i<E.nrJucatori;i++)
      .....
```

3. În secvența de instrucțiuni de mai jos variabilele **i** și **j** sunt de tip întreg, iar variabila **A** memorează un tablou bidimensional cu 5 linii și 5 coloane, numerotate de la 0 la 5. Elementele tabloului sunt numere întregi. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, tabloul memorat în variabila **A** să aibă elementele din figura alăturată. **(6 puncte)**

```
for(i=0;i<6;i++)
  for(j=0;j<=2;j++)
    .............
```

```
0 1 2 2 1 0
1 2 3 3 2 1
2 3 4 4 3 2
2 3 4 4 3 2
1 2 3 3 2 1
0 1 2 2 1 0
```


SUBIECTUL III (30 de puncte)

1. Scrieți un subprogram C/C++, **divprim** cu trei parametri:
– n, prin care primește un număr natural nenul de cel mult 5 cifre;
– k, prin care primește numărul de elemente ale vectorului v;
– v, vector de cel mult 100 de numere naturale nenule, ordonate crescător.
Subprogramul va determina cel mai mic divizor prim al lui n și îl va căuta în vectorul v. Dacă divizorul prim nu va fi găsit, acesta va fi adăugat în v pe poziția corespunzătoare astfel încât ordinea crescătoare să se păstreze. Valorile modificate ale vectorului și dimensiunii acestuia vor fi furnizate tot prin parametrii k și v.
Exemplu: Dacă la apel funcția va primi ca parametri n = 27, k = 4 și v = (2, 5, 11, 17), după execuția acesteia k = 5, iar v = (2, 3, 5, 11, 17). **(10 puncte)**

2. Numim **dublu-cuvânt** un cuvânt cu număr par de litere care este format prin dublarea unui cuvânt. De exemplu, `tictic` este un dublu-cuvânt, în timp ce `tictac` sau `abba` nu sunt. Scrieți un program C/C++ care citește de la tastură S, un text de lungime maximă 255 de caractere, format din cuvinte separate prin câte un spațiu și modifică în memorie șirul, înlocuind a doua jumătate a fiecărui dublu-cuvânt cu caracterul *. Dacă s-a făcut cel puțin o înlocuire, se va afișa pe ecran șirul rezultat, altfel se va afișa mesajul `NEMODIFICAT`.
Exemplu: Dacă se citește: `se aude tictic tictac apoi dingding dingding`
se va afișa: `se aude tic* tictac apoi ding* ding*` **(10 puncte)**

3. Fișierul text **bac.txt** conține cel mult 10^6 numere naturale de cel mult 9 cifre, separate prin câte un spațiu. Se cere afișarea pe ecran a valorii k pentru care intervalul de forma $[2^k, 2^{k+1})$ conține cele mai multe numere din șir. Dacă există mai multe astfel de intervale, se va afișa cel care are k maxim. Proiectați un algoritm eficient din punct de vedere al timpului de executare.
Exemplu: Dacă fișierul bac.txt conține numerele
`175 22 2018 512 1025 18` atunci pe ecran se afișează: `10`, deoarece sunt 2 intervale cu câte 2 numere: $22, 16 \in [2^4, 2^5)$, $2018, 1025 \in [2^{10}, 2^{11})$, dar al doilea interval are k maxim.
a) Descrieți în limbaj natural algoritmul utilizat, justificând eficiența acestuia. **(2 puncte)**
b) Scrieți programul C/C++ corespunzător algoritmului descris la punctul **a)**. **(8 puncte)**

Teza 11

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Variabilele a, b, c, d sunt întregi, a ≤ b, c ≤ d. Care dintre expresiile de mai jos are valoarea 1 dacă și numai dacă intersecția intervalelor (a, b) și [c, d] este vidă?
   a) `c ≥ b && a < d`
   b) `! (d > a || c ≤ b )`
   c) `! (b > c && a < d)`
   d) `a ≥ d || b < c`

2. Care este numărul de subgrafuri euleriene cu 3 vârfuri care se pot construi dintr-un graf neorientat cu 6 vârfuri și 6 valori egale cu 0 în matricea de adiacență corespunzătoare?
   a) 30 **b)** 6 **c)** 15 **d)** 20

3. Se utilizează metoda backtracking pentru a genera în ordine lexicografică, toate anagramele distincte ale cuvântului **`memorie`** astfel încât să nu conțină litere identice pe poziții alăturate și litera o să apară după litera m, nu neapărat consecutiv. Primele soluții generate sunt **`eiemrmo, eimemor, eimrmio, eimrmoi.`** Care sunt penultima și ultima soluție generată?
   a) `rmiemoe, rmimeoe`
   b) `rimemeo, rimemoe`
   c) `mrmieoe, mrmoeie`
   d) `mirmeoe, rmiemoe`

4. Într-un arbore cu 100 de noduri, fiecare nod care nu este frunză are cel mult 2 fii. Care este numărul maxim de frunze ale acestuia?
   a) 100 **b)** 51 **c)** 50 **d)** 99

5. Fie subprogramul F de mai jos.

```
int F (int n, int d)
{ if(d*d<n)
     return (n%d==0 ? d+n/d+F(n,d+1): F(n,d+1));
  else return d*d==n ? d: 0;}
```

   Indicați apelul care determină calcularea sumei tuturor divizorilor proprii pozitivi ai numărului 100 (divizori diferiți de 1 și de 1 000).
   a) F(100,1) **b)** F(100,2) **c)** F(2,100) **d)** F(1,100)

SUBIECTUL II (40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

a) Scrieți valoarea care se va afișa pentru n = 36. **(6 puncte)**
b) Scrieți cel mai mic număr de trei cifre ce poate fi citit pentru n astfel încât rezultatul afișat să fie 0. **(6 puncte)**
c) Scrieți programul C/C++ corespunzător algoritmului dat. **(10 puncte)**
d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se utilizeze alt tip de structură repetitivă. **(6 puncte)**

```
citește n (n număr natural nenul)
k←0
a←1
cât timp a * a ≤ n execută
    dacă  n % a=0 atunci
     b←[n /a]
     dacă  a % 2=b % 2 atunci
        k←k + 1
    a←a+1
scrie k
```

2. Se consideră declarările C/C++ de mai jos: **Fisier**, care memorează numele, extensia și dimensiunea în octeți ale unui fișier de pe hard-disc și **Dosar**, care memorează numărul de fișiere dintr-un dosar (cel mult 100) și descrierea lor conform structurii Fisier. Scrieți o secvență de instrucțiuni C/C++ care atribuie variabilei sum dimensiunea totală a fișierelor din dosarul memorat în D, al căror nume începe cu litera A. **(6 puncte)**

```
struct Fisier {
 char nume[30],extensie[5];
 int dimensiune;};
struct Dosar {
  Fisier F[100];
  int nrFisiere;
} D;
int sum, i;
```

3. Ce se va afișa în urma executării secvenței de program de mai jos? **(6 puncte)**

```
char s[20]="bacalaureat", t[20]="mate-informatica";
s[3]=s[6]=' ';
strcpy(s+7,strchr(t,'i'));
cout<<s;
```

SUBIECTUL III (30 de puncte)

1. Scrieți un program C/C++ care citește de la tastatură un număr natural N (N∈[2,50]), multiplu de 3 și elementele unui tablou bidimensional cu cifre binare dispuse pe N linii și N coloane, numerotate de la 0 la N – 1. Programul modifică tabloul în memorie înlocuind fiecare linie formată doar din valori 0 cu valorile reprezentând numărul de 1 de pe coloana corespunzătoare din tabloul inițial apoi afișează rezultatul pe ecran. **(10 puncte)**

Exemplu: Dacă N=5 și tabloul este:

```
1 0 0 1 1
1 1 1 0 1
0 0 0 0 0
1 1 0 1 1
0 0 0 0 0
```

atunci se obține tabloul următor:

```
1 0 0 1 1
1 1 1 0 1
3 2 1 2 3
1 1 0 1 1
3 2 1 2 3
```

2. Subprogramul **nPrime** are trei parametri:

- **n**, prin care primește un număr natural ($2 \le n \le 10^3$);
- **k**, prin care furnizează numărul de valori prime cu n, mai mici decât n;
- **p**, prin care furnizează șirul celor k valori prime cu n, în ordine crescătoare. Spunem că *x este prim cu n* dacă cele două numere nu au divizori comuni mai mari decât 1.

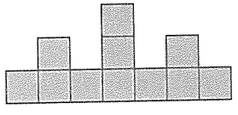
Scrieți în limbajul C/C++ definiția completă a subprogramului **nPrime**.

Exemplu: Dacă **n=20** atunci **k=8**, iar **p=(1,3,7,9,11,13,17,19).** **(10 puncte)**

3. Într-un depozit au fost așezate cutii identice, una după alta, eventual suprapuse, astfel încât numărul maxim de cutii suprapuse într-o stivă este n ($0 < n \le 15$), iar între două stive cu același număr de cutii să existe cel puțin una cu mai multe cutii decât oricare dintre cele două. Considerăm că o stivă poate fi formată dintr-o singură cutie.

Scrieți un program care citește din fișierul text **bac.txt** o valoare n, reprezentând numărul maxim de cutii care pot fi stivuite una peste alta și afișează pe ecran numărul maxim de stive care se pot construi respectând regulile date și numărul de cutii ce ar putea fi aranjate astfel. Proiectați un algoritm eficient din punct de vedere al timpului de executare și al spațiului de memorie utilizat.

Exemplu: Dacă n = 3, atunci numărul maxim de stive = 7 și numărul total de cutii = 11, distribuite astfel:



a) Realizați o descriere de 3-4 rânduri a algoritmului ales justificându-se eficiența acestuia. **(2 puncte)**

b) Scrieți programul C/C++ corespunzător metodei descrise la punctul **a)**. **(8 puncte)**

Teza 12

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Știind că x este o variabilă care memorează un număr întreg nenul, precizați care este cea mai mică valoare ce se poate obține la evaluarea următoarei expresii C/C++: `200%x - x%5`.

a) -203 **b)** 0 **c)** -195 **d)** -4

2. Care este numărul de valori egale cu 0 din matricea de adiacență a unui graf eulerian cu 10 vârfuri și număr maxim de muchii?

a) 20 **b)** 10 **c)** 15 **d)** 50

3. Se generează în ordine crescătoare toate numerele naturale de 4 cifre alese din mulțimea **3, 7, 2, 0, 5, 8** astfel încât cifrele de pe poziții alăturate să aibă parități diferite. Știind că primele 5 numere generate sunt **2305, 2307, 2385, 2387**, care dintre următoarele secvențe reprezintă o secvență de numere generate unul după altul:

a) 5087, 5237, 5238 **c)** 7850, 7852, 8305
b) 5087, 5237, 5238 **d)** 7830, 7832, 7835

4. Fie următoarea secvență de numere: 4, 3, 3, 1, 1, 1, 1, 1, 1. Care dintre următoarele șiruri poate fi vectorul de tați ai unui arbore cu secvența gradelor vârfurilor dată mai sus și înălțime maximă?

a) (2,3,4,5,6,0,6,7,6) **b)** (0,1,1,1,2,2,9,9,1)
c) (9,1,1,1,2,2,3,3,0) **d)** (4,1,1,0,6,2,6,2,6)

5. Se consideră subprogramul f definit mai jos.

```
void F(int a, int b)
{ if (b > a)
  {  cout << b;
     F(a, b/2);
  }
  else if (b < a)
       {  F(a/2, b);
          cout << a;
       }
       else cout << '#';
  }
```

Ce se va afișa în urma apelului F(7,8)?

a) 84237# **b)** 842#37 **c)** 8#4237 **d)** 8#3742

SUBIECTUL II (40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

a) Ce se va afișa dacă a = 8 și b = 12? **(6 puncte)**

b) Pentru a = 25, care este cea mai mare valoare pe care o poate primi b astfel încât să se afișeze doar '*'? **(6 puncte)**

c) Scrieți programul C/C++ corespunzător algoritmului dat. **(10 puncte)**

d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască „cât timp... execută" cu alt tip de structură repetitivă. **(6 puncte)**

```
citește a, b (a, b numere naturale, a≤b)
y←0
pentru x←b, a, –1 execută
    y←x
    k←0
    cât timp y>0 execută
        k←k+1–2*(y%2)
        y←[y/2]
    dacă k=0 atunci
        scrie x, ' '
scrie '*'
```

2. Se consideră declarările C/C++ de mai jos, în care variabilele `T` memorează 100 de turnuri de forma paralelipiped, pentru care sunt cunoscute lungimea, lățimea, înălțimea și culoarea. Scrieți o secvență de instrucțiuni C/C++ care afișează culorile turnurilor care au formă de cub și sunt memorate în variabila `T`. **(6 puncte)**

```
struct Turn {
   int lungime, latime, inaltime;
   char culoare[20];};
Turn  T[100];
int i;
```

3. În secvența de instrucțiuni de mai jos variabilele **i** și **j** sunt de tip întreg, iar variabila A memorează un tablou bidimensional cu 6 linii și 6 coloane, numerotate de la 1 la 6. Elementele tabloului sunt numere întregi. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, tabloul memorat în variabila A să aibă elementele din figura de mai jos. **(6 puncte)**

```
for(i=1;i<=6;i++)
  for(j=1;j<=6;j++)
    ............
```

```
2 3 1 2 3 1
3 6 2 3 6 2
1 2 0 1 2 0
2 3 1 2 3 1
3 6 2 3 6 2
1 2 0 1 2 0
```


SUBIECTUL III (30 de puncte)

1. Scrieți un program C/C++ care citește de la tastatură un șir de caractere format din cel mult 250 de caractere, litere mici ale alfabetului englez și afișează pe ecran șirul rezultat și numărul minim de etape de eliminare astfel încât să nu mai existe în șir două caractere alăturate identice. O etapă de eliminare se aplică unei secvențe de caractere identice care începe la poziția i din șir, șterge secvența de la poziția i, apoi repetă ștergerea secvențelor de caractere identice ce includ poziția i, până cînd nu se mai obțin caractere identice alăturate.
Exemplu: Dacă se citește șirul `mtopppppootatnnnne` se va afișa `mate 2`, pentru că sunt 2 etape de eliminare: `toppppoot` și `nnnn`. **(10 puncte)**

2. Scrieți un subprogram C/C++, denumit **perechi**, care primește prin parametrii `n` și `m` două numere naturale, nenule, de cel mult două cifre. Subprogramul va furniza prin parametrul `k` numărul de perechi de numere `x, y (x<y<=n)`, care au cel mai mic multiplu comun `m`. Perechile determinate se vor afișa pe ecran, pe linii diferite, separate prin câte un spațiu.
Exemplu: Pentru n = 6 și m = 6 se afișează perechile:

```
1 6
2 3
2 6
3 6, iar în k va fi valoarea 4.
```

(10 puncte)

3. Scrieți un program C/C++ care citește din fișierul text **BAC.TXT** un șir cu cel mult 1 000 000 de numere întregi formate dintr-o cifră fiecare și afișează pe ecran separate printr-un spațiu, numărul sau numerele din fișier cu număr maxim de apariții.
Exemplu: Dacă în fișier sunt numerele `1 -8 1 1 -8 -8 9 9` atunci pe ecran se vor afișa nu neapărat în această ordine `-8 1`.

a) Se cere să se proiecteze un algoritm eficient din punct de vedere al timpului de executare și al spațiului de memorie utilizat și să se realizeze o descriere de 3-4 rânduri a algoritmului ales justificându-se eficiența acestuia. **(2 puncte)**

b) Scrieți programul C/C++ corespunzător metodei descrise la punctul **a)**. **(8 puncte)**

Teza 13

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Variabilele **x, y** și **z** sunt de tip întreg. Care dintre următoarele expresii C/C++ are valoarea 1 dacă și numai dacă cel mult două dintre variabilele precizate au valori egale?

a) `!(x==y && x==z)||y!=z` **c)** `(x-y)*(x-z)*(y-z)==0`

b) `x==y || x==z || y==z` **d)** `x==y==z`

2. Care este înălțimea maximă a unui arbore cu 50 de noduri în care fiecare nod care nu este frunză are mai mulți fii decât părintele său?

a) 25 **b)** 10 **c)** 9 **d)** 11

3. Se generează prin metoda backtracking mulțimi distincte cu elemente numere naturale nenule și cu proprietatea că suma elementelor fiecărei mulțimi este egală cu 7 astfel: `{1,2,4},{1,6},{2,5},{3,4},{7}`. Folosind aceeași metodă pentru a genera mulțimi distincte cu elemente numere naturale nenule și cu proprietatea că suma elementelor fiecărei mulțimi este egală cu 9, stabiliți în ce ordine sunt generate următoarele mulțimi:
`M1= {2, 3, 4}; M2={3, 6}; M3={2, 7}; M4={1, 8}.`

a) M4, M1, M3, M2 **c)** M1, M3, M2, M4

b) M4, M1, M2, M3 **d)** M1, M2, M3, M4

4. **Distanța** dintre două noduri într-un graf neorientat este lungimea minimă a unui lanț cu extremitățile în cele două noduri. **Ponderea unui nod** este suma distanțelor de la acesta la toate celelalte noduri din graf. Care poate fi ponderea maximă a unui nod într-un graf neorientat cu 10 noduri?

a) 20 **b)** 90 **c)** 10 **d)** 45

5. Se consideră subprogramul F definit de mai jos.

```
void F()
{  int x; cin>>x;
   if (x>0) { cout<<x<<' ';F();}
  else if (x<0) { F(); cout<<x<<' ';}
}
```

Ce se va afișa la citirea următorului șir de valori: `4 -5 -2 8 7 0`?

a) `7 8 4 -2 -5` **c)** `-5 -2 4 8 7`

b) `4 8 7 -5 -2` **d)** `4 8 7 -2 -5`

SUBIECTUL II (40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

```
citește n (n număr natural)
x←0; y←1;
m←[n/2];
┌cât timp m>0 execută
│ x←x+y
│ y←y+x
│ m←m–1
└■
z ← (1–n%2)* x+(n%2)*y
scrie z
```

a) Scrieți valoarea care se va afișa pentru n = 7. **(6 puncte)**

b) Scrieți cea mai mică valoare care se poate citi pentru n astfel încât rezultatul afișat să fie un număr de trei cifre. **(6 puncte)**

c) Scrieți programul C/C++ corespunzător algoritmului dat. **(10 puncte)**

d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască structura repetitivă cât timp... execută cu o structură repetitivă cu condiție finală. **(6 puncte)**

2. Variabila C memorează un cerc descris prin raza R, un număr natural nenul și O, un punct de coordonate reale a, b reprezentând abscisa și ordonata centrului acestuia. Se consideră declarările C/C++ corespuzătoare descrierii de mai jos.

```
struct Punct {
  float x,y;};
struct Cerc {
  Punct centru;
  int raza;} C;
```

Scrieți o expresie C/C++ care să aibă valoarea 1 dacă și numai dacă cercul C are centrul pe axa Ox și intersectează axa Oy. **(6 puncte)**

3. În secvența de instrucțiuni de mai jos variabilele **i** și **j** sunt de tip întreg, iar variabila A memorează un tablou bidimensional cu 5 linii și 5 coloane, numerotate de la 1 la 5. Elementele tabloului sunt numere întregi. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, tabloul memorat în variabila A să aibă elementele din figura de mai jos. **(6 puncte)**

```
for(i=5;i>=1;i--)
  for(j=1;j<6;j++)
    ..............
```

```
5 1 1 1 2
5 2 0 0 1
5 1 2 0 1
5 2 1 2 1
5 5 5 5 5
```

SUBIECTUL III (30 de puncte)

1. Se consideră un text cu cel mult 200 de caractere, litere, cifre și spații. Cuvintele sunt formate din litere ale alfabetului englez și/sau cifre și sunt separate prin câte un spațiu. Scrieți un program C/C++ care citește de la tastatură un text de tipul menționat mai sus și construiește în memorie apoi afișează pe ecran șirul format din prima literă a fiecarui cuvânt care nu conține nicio cifră, transformată în majusculă. Dacă nu există astfel de cuvinte, se afișează pe ecran mesajul **nu există**.
Exemplu: Pentru șirul `Varianta T100 data in 10nov la simulare 12C` se afișează `VDILS`. **(10 puncte)**

2. Un număr n se numește **rotund** dacă este par și în reprezentarea sa matematică în baza 2, nu sunt două cifre egale alăturate. De exemplu, numărul `10` este un număr rotund deoarece reprezentarea sa în baza 2 este `1010`. Numărul `18` nu este rotund deoarece reprezentat în baza 2 este `10010`.
Scrieți definiția completă a unui subprogram **rotund**, care primește prin intermediul parametrului `n` un număr natural nenul de cel mult 9 cifre și returnează 1 dacă n este număr rotund și 0 altfel. **(10 puncte)**

3. Fiind date două numere a și b, îl numim pe **a sufix al lui b** dacă a este egal cu b sau dacă b se poate obține din a prin alipirea la stânga a unor noi cifre. Exemplu: 12 este sufix al lui 12, iar 15 este sufix al lui 31415. Fișierul **bac.txt** conține pe prima linie un număr natural x, cu cel mult 4 cifre, iar pe a doua linie un șir de cel puțin două și cel mult 1 000 000 de numere naturale cu cel mult nouă cifre. Numerele din șir sunt separate prin câte un spațiu.
Se cere să se afișeze pe ecran ultimul termen al șirului care are ca sufix numărul x. Dacă în șir nu există o astfel de valoare, pe ecran se afișează mesajul **nu exista**. Pentru determinarea numărului cerut se utilizează un algoritm eficient din punctul de vedere al memoriei și al timpului de executare.
Exemplu: Dacă fișierul **bac.txt** conține numerele
`12`
`3445 89312 1212 12 67120 312 1234578` atunci pe ecran se afișează `312`.

a) Descrieți în limbaj natural algoritmul utilizat, justificând eficiența acestuia. **(2 puncte)**
b) Scrieți programul C/C++ corespunzător algoritmului descris la punctul **a)**. **(8 puncte)**

Teza 14

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Variabilele **a, b** și **c** memorează numere naturale nenule. Care dintre expresiile C/C++ următoare este 1 dacă și numai dacă c este un multiplu comun al valorilor memorate în a și b?
a) `a%c+b%c==0` **c)** `!(a%c==0 || c%b==0)`
b) `(c%a)*(c%b)==0` **d)** `c%a+c%b==0`

2. Un graf orientat cu 5 vârfuri, numerotate de la 1 la 5, are arcele `(1,4), (1,5), (2,1), (2,3), (3,1), (3,4), (4,2), (5,1), (5,3).` Numărul maxim de arce care se pot elimina, astfel încât graful parțial obținut să fie tare conex este:
a) 2 **b)** 3 **c)** 4 **d)** 5

3. Cu ajutorul metodei backtracking se generează în ordine lexicografică, toate parolele formate din 5 caractere distincte din mulțimea `{'a','b','c','e','f','g','u'}` și care conțin cel puțin 2 vocale și cel puțin 2 consoane. Primele soluții generate sunt `abcef, abceg, abceu, abecf`. Care dintre șirurile următoare NU reprezintă secvență de șiruri generate consecutiv?
a) `fegua, fegub, feguc` **c)** `bacue, bacuf, bacug`
b) `gufab, gufeb, gufec` **d)** `efabu, efacb, efacg`

4. Se consideră un graf neorientat conex cu 100 de muchii. Numărul maxim de noduri din graf este:
a) 100 **b)** 99 **c)** 101 **d)** 50

5. Se consideră subprogramul F definit mai jos.
```
long F(int x, int y)
{ if (x==y || x==0)
    return 1;
  else{ y--;
    return F(x,y) + F(x-1,y);}
}
```
Ce se va afișa în urma apelului `F(7,10)`?
a) 120 **b)** 100 **c)** 75 **d)** 34

SUBIECTUL II (40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

a) Scrieți valoarea care se va afișa dacă se citesc în ordine numerele 6 3 –2 5 –4 –8 1. **(6 puncte)**

b) Dacă pentru n se citește valoarea 4, scrieți un șir de 4 valori distincte astfel încât rezultatul afișat să fie 0. **(6 puncte)**

c) Scrieți programul C/C++ corespunzător algoritmului dat. **(10 puncte)**

d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască structura repetă „cât timp... execută" cu o structură repetitivă „pentru". **(6 puncte)**

```
citește n (n număr natural nenul)
k←1; p←0
┌cât timp n > 0 execută
  citește x (x, număr întreg)
  ┌dacă  x < 0 și p > x atunci
      p←x
  altfel
      ┌dacă x<0 && p<=x atunci
          k←0
      └■
  └■
  n←n – 1
└■
scrie k
```

2. Considerăm că un produs este descris prin cod (număr natural), luna și anul expirării (numere naturale) și prețurile a 5 posibili furnizori (numere reale). Scrieți declararea C/C++ a unei structuri cu eticheta **Produs** și a unei variabile **P** capabilă să memoreze 100 de produse, astfel încât construcțiile de mai jos să fie corecte sintactic. **(6 puncte)**

```
P[0].cod
P[0].data_expirarii.luna,   P[0].data_expirarii.an
P[0].pret_furnizor[1]
```

3. În secvența de instrucțiuni de mai jos variabilele **i** și **j** sunt de tip întreg, iar variabila **A** memorează un tablou bidimensional cu 7 linii și 7 coloane, numerotate de la 1 la 7. Elementele tabloului sunt caractere. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, tabloul memorat în variabila **A** să aibă elementele din figura de mai jos. **(6 puncte)**

```
for(i=0;i<=6;i++)
  for(j=0;j<=6;j++)
     ..............
```

```
* - - - - - *
* * - - - * *
* - * - * - *
* - - * - - *
* - - - - - *
* - - - - - *
* - - - - - *
```


SUBIECTUL III (30 de puncte)

1. Operația **WRAP de dimensiune** n, aplicată asupra unui text format din cuvinte separate prin spații, va încadra textul pe un număr minim de linii de n caractere, astfel încât orice cuvânt va fi afișat complet pe linia sa. Dacă la sfârșitul liniei rămân caractere ce nu pot fi ocupate de un cuvânt ele se vor înlocui cu *. Dacă un spațiu din text ajunge pe prima poziție din linie, el se va ignora, deoarece nicio linie nu poate începe cu spațiu.
Scrieți un program C/C++ care citește de la tastatură, de pe prima linie, un număr natural n ($10 < n < 50$), iar de pe a doua linie, un text S de cel mult 200 de caractere (litere mici ale alfabetului englez și spații), format din cuvinte separate prin câte un spațiu. Textul nu începe și nu se termină cu spații. Programul afișează pe ecran liniile paragrafului în care se încadrează textul după operația WRAP de dimensiune n. Dacă afișarea nu e posibilă se va scrie pe ecran mesajul IMPOSIBIL. **(10 puncte)**
De exemplu, dacă se citește **n = 20** și textul:

```
bacul la info este usor daca esti foarte bine pregatit
se va afișa
bacul la info este *
usor daca esti *****
foarte bine pregatit
```

2. Scrieți în limbajul C/C++ definiția completă a subprogramului **regulat** care primește ca parametru un număr natural nenul n, de cel mult 5 cifre și returnează cel mai mare număr regulat, mai mic sau egal cu n. Un număr natural nenul se numește **regulat** dacă poate fi scris sub forma $2^A * 3^B * 5^C$, unde A, B, C sunt numere naturale.
Exemplu: Pentru **n = 44**, subprogramul va returna după apel valoarea **40**, $40 = 2^3 * 3^0 * 5^1$. **(10 puncte)**

3. Din fișierul text **BAC.TXT** se citesc, de pe prima linie un număr k ($k < 10^6$), iar de pe linia următoare cel mult 1 000 000 de numere naturale cu cel mult 2 cifre fiecare, separate prin câte un spațiu. Să se afișeze pe ecran numărul de pe a doua linie din fișier care va fi pe poziția k în șirul ordonat descrescător, dacă există sau –1 în caz contrar. Să se proiecteze un algoritm eficient din punct de vedere al timpului de executare.
Exemplu: Dacă fișierul conține numerele:
6
21 9 16 2 16 2 9 4 9 21 9 2 8 atunci pe ecran se va afișa 9.
a) Realizați o descriere de 3-4 rânduri a algoritmului ales justificând eficiența acestuia. **(2 puncte)**
b) Scrieți programul C/C++ corespunzător metodei descrise la punctul **a)**. **(8 puncte)**

Teza 15

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Variabila a memorează un număr natural cu 4 cifre. Care dintre expresiile C/C++ următoare inserează cifra 0 în numărul n, pe poziția cifrei sutelor?
   a) `n=(n/100)*10000 + n%100`
   b) `n=n%10+n/10*10+n/100*1000`
   c) `n=n/10%100*1000+n/100`
   d) `n=n%100+1000*n/100`

2. Un graf orientat cu 5 noduri numerotate de la 1 la 5, este dat prin listele de adiacență de mai jos.
   L1: 2,5 L2: 5 L3: 2 L4: 1 L5: 1,2,4
   Care este numărul minim de arce ce trebuie adăugate astfel încât graful orientat să fie complet?
   a) 2 **b)** 12 **c)** 5 **d)** 10

3. Se generează cu ajutorul metodei backtracking, în ordine crescătoare, numere formate doar din cifrele 0, 1 și 2, care au suma cifrelor egală cu 2. Primele 5 numere, în ordinea în care au fost generate, sunt: 2, 11, 20, 101, 110. Dacă se generează după aceeași regulă numere formate doar din cifrele 0, 1, 2 și 3, care au suma cifrelor egală cu 4, care vor fi primele 3 astfel de numere scrise?
   a) 13, 31, 103, 130, 301
   b) 13, 22, 31, 103, 112
   c) 4, 13, 22, 31, 40
   d) 13, 22, 31, 40, 103

4. Într-un graf neorientat cu 10 de vârfuri, suma gradelor vârfurilor este 40. Care este numărul maxim de componente conexe dintr-un astfel de graf?
   a) 5 **b)** 4 **c)** 6 **d)** 1

5. Se consideră subprogramul f definit alăturat:

```
void F(int a, int b){
       for(int i=a; i<=b; i++){
            F(i+1, b);
            cout<<i;
        }
  }
```

   Care instrucțiune de apel trebuie executată astfel încât în urma apelului să se afișeze F(1,3)?
   a) 1323323 **b)** 32313 **c)** 32123 **d)** 3231323

SUBIECTUL II (40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

```
citește n (n număr natural, n>1)
p←1; m←0; x←2
┌cât timp n >= x execută
│ ┌dacă n%x=0 atunci
│ │   m←m+1; n←[n/x]
│ altfel
│ │ ┌dacă m%2 ≠ 0 atunci
│ │ │   p←p*x
│ │ └
│ │   m←0; x←x+1
│ └
└
┌dacă m%2 ≠ 0 atunci
│   p←p*x
└
scrie p
```

   a) Scrieți valoarea care se va afișa pentru n = 23100. **(6 puncte)**
   b) Scrieți cel mai mic și cel mai mare număr de două cifre care poate fi citit pentru n astfel încât algoritmul să afișeze valoarea 1. **(6 puncte)**
   c) Scrieți programul C/C++ corespunzător algoritmului dat. **(10 puncte)**
   d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască structura repetitivă „cât timp... execută" cu o structură repetitivă cu condiție finală. **(6 puncte)**

2. Pentru memorarea rezultatelor la examenul de BAC obținute de un candidat se consideră declarările C/C++ de mai jos.

```
struct rezBac {
  int mat,rom,inf;
  char rez;
} candidat;
```

Scrieți o instrucțiune C/C++ care să atribuie litera **'A'** rezultatului elevului **candidat**, dacă fiecare dintre notele la probele de română, mate și info sunt cel puțin 5, iar media lor este cel puțin 6, iar în caz contrar va atribui litera **'R'**. **(6 puncte)**

3. În secvența de instrucțiuni de mai jos variabilele **i** și **j** sunt de tip întreg, iar variabila **A** memorează un tablou bidimensional cu 6 linii și 6 coloane, numerotate de la 1 la 6. Elementele tabloului sunt numere întregi. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, tabloul memorat în variabila **A** să aibă elementele din figura de mai jos. **(6 puncte)**

```
for(i=1;i<=3;i++)
  for(j=1;j<=3;j++)
    ..............
```

```
1 1 1 1 1 1
1 2 2 2 2 1
1 2 3 3 2 1
1 2 3 3 2 1
1 2 2 2 2 1
1 1 1 1 1 1
```

SUBIECTUL III (30 de puncte)

1. Scrieți un program C/C++ care citește de la tastatură un șir S, format din cel mult 100 de caractere, litere mici ale alfabetului englez. Programul modifică în memorie șirul S, adăugând la sfârșit un număr minim de caractere astfel încât S să fie palindrom (parcurs de la stânga la dreapta și de la dreapta la stânga, să se obțină același șir), apoi afișează pe ecran șirul obținut după transformare.
   Exemplu: Dacă se citește șirul **repetate** atunci se va afișa șirul **repetateper,** iar dacă se citește șirul **info** atunci se va afișa șirul **infoofni.** **(10 puncte)**

2. Subprogramul **compact** primește prin parametrul `n`, un număr natural ($2 \le n \le 1000$) și prin parametrul `a`, un tablou unidimensional care memorează un șir de `n` numere naturale, fiecare cu cel mult 2 cifre. Subprogramul elimină din tablou, pentru fiecare a_i, $1 \le i \le n$, elementele egale cu acesta, situate pe poziții consecutive, astfel încât, tabloul să conțină la final, un număr maxim de elemente distincte, în ordinea în care au apărut în șirul primit ca parametru. Șirul modificat și numărul de elemente rămase după eliminare vor fi furnizate tot prin intermediul parametrilor `n` și `a`. Scrieți în limbajul C/C++ definiția completă a subprogramului **compact**.
   Exemplu: Dacă `n=9` și tabloul memorează valorile `1 2 2 2 3 3 5 8 9` atunci, după apel `n=6`, iar tabloul va memora `1 2 3 5 8 9`. **(10 puncte)**

3. Un șir de numere $x_1, x_2, \ldots x_k, \ldots x_m$ conține o **secvență – V** dacă are un element x_k $(1<k<m)$ și există **i** și **p** $(1 \le i < k < p \le m)$ cu proprietatea: $x_i > x_{i+1} > \ldots > x_k$ și $x_k < x_{k+1} < \ldots < x_p$. Scrieți un program care citește din fișierul **bac.txt** cel mult 1 000 000 numere întregi, având cel mult nouă cifre și afișează pe ecran lungimea maximă a unei secvențe – V și numărul de secvențe – V de lungime maximă pentru șirul de numere din fișier sau afișează mesajul **Nu există**, dacă nu există nicio secvență – V în șir. Elaborați un algoritm eficient ca timp de execuție și spațiu de memorie utilizat.
   Exemplu: Pentru n = 14, șirul de numere `9,8,2,11,14,10,9,20,19,14,9,20,56,17` conține două secvențe – V cu lungimea maximă egală cu 5 și se va afișa pe ecran 5 2.
   a) Descrieți în limbaj natural algoritmul proiectat și justificați eficiența acestuia. **(2 puncte)**
   b) Scrieți programul C/C++ corespunzător algoritmului descris. **(8 puncte)**

Teza 16

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Variabilele a și b memorează numere întregi. Care dintre următoarele expresii C/C++ are valoarea 1 dacă și numai dacă a și b sunt numere impare, de semne contrare?
   a) `a%2+b%2==2 && a<0 && b>0`
   b) `a*b<0 && (a*b)%2!=0`
   c) `(a<0 && b>0||a>0 && b<0) && a%2!=b%2`
   d) `a%2 && b%2 && a+b<0`

2. Un graf neorientat cu 10 noduri este format din 3 componente conexe, fiecare subgraf complet. Care dintre afirmațiile următoare sunt adevărate?
   A1. gradul maxim al unui nod poate fi 7 **A2.** graful nu poate conține vârfuri izolate
   A3. suma gradelor vârfurilor e impară **A4.** graful are cel puțin 12 muchii
   a) A1 și A2 **b)** A2, A3 și A4 **c)** A1 și A3 **d)** A2 și A4

3. Se consideră generate cu ajutorul metodei backtracking toate rezultatele posibile ale unei serii de 10 aruncări cu banul care produce secvența $R_1R_2 \ldots R_{10}$ unde $R_k \in \{C, P\}$ este rezultatul aruncării k (C – cap, P – pajură). Câte secvențe vor avea același număr de capete și de pajuri?
   a) 1024 **b)** 512 **c)** 252 **d)** 32

4. Într-un graf orientat cu 7 vârfuri, numerotate de la 1 la 7, pentru oricare două vârfuri ale sale **i** și **j** $(i \ne j)$, există arcul **(i, j)** fie dacă **i** este multiplu al lui **j,** fie dacă **i** și **j** au aceeași paritate, iar **i<j**. Care sunt vârfurile pentru care gradul interior este mai mare sau egal cu cel exterior?
   a) 1, 5, 7 **b)** 2, 3, 6 **c)** 1, 7 **d)** 4, 5

5. Se consideră subprogramele F și G definite mai jos.

```
int F(int a){
     if(a%2==0) return F(a/2);
     else return a;
}
void G(int n)
{
     if (n<1) return 1;
     else return F(n*G(n-1));
}
```

   Ce se valoare se obține prin apelul G(6)?
   a) 120 **b)** 720 **c)** 24 **d)** 45

SUBIECTUL II (40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

```
citește n, x (n, x numere naturale)
pentru i←1, n – 1 execută
    citește y (x,y numere întregi, x ≤ y)
    dacă x ≤ 99 și y ≤ 10 atunci
        k←k+1
    x←y
scrie k
```

a) Scrieți valoarea care se va afișa dacă se citesc în ordine numerele 5 8 9 35 256 1000. **(6 puncte)**

b) Pentru n = 4, scrieți 4 valori distincte ce pot fi citite în x și y astfel încât rezultatul afișat să fie 0. **(6 puncte)**

c) Scrieți programul C/C++ corespunzător algoritmului dat. **(10 puncte)**

d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască structura „pentru... execută" cu o structură repetitivă cu condiție inițială. **(6 puncte)**

2. Considerăm că o campanie de reduceri este descrisă prin titlu (maxim 30 de caractere), luna și ziua de început, durata exprimată în zile consecutive și procentul de discount (numere naturale nenule). Scrieți declararea C/C++ a unei structuri cu eticheta **Campanie** și a unei variabile **C**, capabilă să memoreze 50 de campanii, astfel încât construcțiile de mai jos să fie corecte sintactic. **(6 puncte)**

```
C[1].titlu
C[1].start.luna,  C[1].start.an
C[1].procent
```

3. În secvența următoare variabilele `n` și `i` sunt de tip întreg, iar variabila `s` permite memorarea unui cuvânt format din cel mult 200 de caractere. Cuvântul citit este format din litere mici. Fără a utiliza alte variabile, scrieți secvența de instrucțiuni care poate înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, afișează, în ordine crescătoare a lungimilor, sufixele lui `s` care încep cu cuvântul **`info`**. **(6 puncte)**

```
cin>>s;
n=.....;
for(i=0;i<n;i++)
{
  ............
}
```


SUBIECTUL III (30 de puncte)

1. Scrieți un program C/C++ care citește de la tastatură un număr natural `N` (`N∈[2,10]`) și elementele unui tablou bidimensional cu `N` linii și `N` coloane, caractere din mulțimea {'X', 'U'}. Programul afișează `DA`, dacă există cel puțin o linie, o coloană sau o diagonală completată doar cu caracterul 'X' și `NU`, altfel. **(10 puncte)**

Exemplu: dacă `N=4` și tabloul este:
X U U X atunci pe ecran se va afișa `DA`
U X X X
X X U X
X U X X

2. Subprogramul **nMax** primește prin parametrul n un număr natural de cel mult 6 cifre și prin parametrul c o cifră nenulă. Subprogramul furnizează tot prin parametrul n numărul maxim care se poate obține inserând în n cifra c. Scrieți în limbajul C/C++ definiția completă a subprogramului **nMax**.

Exemplu: Dacă **n = 57332** și **c = 4** atunci, după apel, subprogramul va returna valoarea **574332**. **(10 puncte)**

3. Numim **interval asociat** unui șir de numere, perechea de numere naturale `a,b` (`a≤b`) cu proprietatea că șirul este format din toate valorile naturale distincte, cuprinse între `a` și `b` și scrise în ordine crescătoare. De exemplu, șirul `4 5 6 7 8` are intervalul asociat `4 8`, iar șirul `1 2 3 5 7 8 9` este format din trei subșiruri ale căror intervale asociate sunt `1 3`, `5 5` și `7 9`.

Scrieți un program C/C++ care citește din fișierul text **BAC.TXT** un șir S cu cel mult 1 000 000 de numere naturale din intervalul $[0,10^3]$ și afișează pe ecran extremitățile intervalului care reprezintă intersecția tuturor intervalelor asociate, de lungime maximă, din care este format șirul dat sau mesajul mulțimea vidă, în cazul în care acestea nu conțin niciun element comun.

Exemplu: Dacă fișierul conține numerele `2 3 4 1 2 3 4 5 6 3 4 5 0 1 2 3` atunci pe ecran vor fi afișate valorile `2 3` deoarece șirul este format din 4 intervale asociate `2 4`, `1 6`, `3 5` și `0 3` a căror intersecție este intervalul `2 3`. Dacă fișierul conține numerele `3 4 5 4 5 6 8` atunci pe ecran se va afișa `multimea vida`.

a) Se cere să se proiecteze un algoritm eficient din punct de vedere al timpului de executare și al spațiului de memorie utilizat și să se realizeze o descriere de 3-4 rânduri a algoritmului ales, justificându-se eficiența acestuia. **(2 puncte)**

b) Scrieți programul C/C++ corespunzător metodei descrise la punctul **a)**. **(8 puncte)**

Teza 17

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Știind că variabila reală x aparține reuniunii intervalelor [3, 7] și (20, 30) care dintre expresiile scrise mai jos, NU are valoarea 1?
   a) !((x<3 ||x>7) && (x>=30 || x<=20))
   b) (x<=7 && x>=3) && (x>20 && x<30)
   c) (x<=7 && x>=3) || (x>20 && x<30)
   d) !(x<3 || x>7) || !(x>=30 ||x<=20)

2. Pentru definiția alăturată a subprogramului P, stabiliți ce se afișează la apelul P(4,1).

```
void P(int n, int i)
{  int j;
   if (i < n)
   {     P(n,i+1);
         for(j = 1; j<= i; j++)
         cout<<j+2;
   }
}
```

   a) 334345 **b)** 112123 **c)** 345343 **d)** 123121

3. Un arbore cu 6 noduri, numerotate de la 1 la 6, este reprezentat prin vectorul de tata = (x, y, 4, z, 0, 1). Ce valoare pot avea numerele x, y, z din acest vector, astfel încât arborele să aibă o singură frunză?
   a) 2, 3, 6 **b)** 1, 3, 4 **c)** 2, 3, 5 **d)** 2, 5, 3

4. Utilizând metoda backtracking, se generează, în ordine descrescătoare, toate numerele formate din 4 cifre pare și care au prima cifră diferită de ultima cifră. Primele cinci numere generate sunt: 8886, 8884, 8882, 8880, 8866. Indicați următoarele două numere care sunt generate după numărul 4806.
   a) 4804, 4802 **b)** 4802, 4800 **c)** 4808, 4820 **d)** 4800, 4688

5. Un graf neorientat are 30 muchii și fiecare nod al grafului are gradul un număr nenul. Doar șase dintre noduri au gradul un număr par, restul nodurilor având gradele numere impare. Care este numărul maxim de noduri pe care poate să le aibă graful?
   a) 36 **b)** 54 **c)** 30 **d)** 60

SUBIECTUL II (40 de puncte)

1. Se consideră declarațiile de mai jos, în care variabila **p** memorează denumirea unui produs și data expirării lui. Variabila **azi** memorează data zilei curente. Scrieți o secvență de instrucțiuni C++ care afișează mesajul **produs expirat,** dacă la data zilei curente produsul a depășit data expirării sau mesajul **produsul nu este expirat,** în caz contrar. **(6 puncte)**

```
struct data
{int zi, luna, an;};
struct produs
{ char denumire[41];
data dexp ;};
produs p; data azi;
```

2. În secvența de instrucțiuni de mai jos variabilele **i** și **j** sunt de tip întreg, iar variabila **A** memorează un tablou bidimensional de tip caracter, cu 5 linii și 5 coloane, numerotate de la 0 la 4. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, tabloul memorat în variabila **A** să aibă elementele din figura de mai jos. **(6 puncte)**

```
for(i=0;i<5;i++)
  for(j=0;j<5;j++)
     ..............
```

```
A B C D E
E A B C D
D E A B C
C D E A B
B C D E A
```

3. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

a) Scrieți ce se va afișa dacă se citesc, în această ordine, numerele 5, 3, 2334, 31, 1535, 330, 75. **(6 puncte)**
b) Scrieți un set de date de intrare care să determine afișarea valorii 1000. **(6 puncte)**
c) Scrieți programul C++ corespunzător algoritmului dat. **(10 puncte)**
d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască structura „pentru" cu alt tip de structură repetitivă. **(6 puncte)**

```
citește n,y (y număr natural, y<10,
n număr natural nenul)
e←0
pentru i←1, n execută
  citește x (număr natural nenul)
  cx←x
  nr←0
  cât timp x≠0 execută
     dacă x%10=y
       atunci
       nr←nr+1
     x←[x/10]
  dacă nr%2=0 și nr>0
    atunci
    e←e+cx
scrie e
```

SUBIECTUL III (30 de puncte)

1. Se citește de la tastatură un șir care are cel mult 250 de caractere: litere și spații. Șirul este format din mai multe cuvinte, separate între ele printr-un spațiu. Scrieți un program C++ care modifică în memorie șirul de caractere, prin interschimbarea caracterelor din fiecare pereche formată din două caractere alăturate, unul fiind vocală și celălalt fiind consoană.
Exemplu: Dacă se citește șirul **exemple de probleme** se va afișa pe ecran șirul **xemplee ed porbelem**. **(10 puncte)**

2. Subprogramul **termen** are un singur parametru **n**, prin care primește un număr natural ($1 \le n \le 100$) și returnează termenul cu numărul de ordine **n** din șirul următor: 1, 2, **3**, 2, 1, 2, 3, 4, 5, **6**, 5, 4, 3, 2, 1, 2, 3, 4, 5, 6, 7, 8, **9**, 8, 7, 6, 5, 4,3, 2, 1 etc. Scrieți în limbajul C/C++ definiția completă a subprogramului **termen**.
Exemplu: Dacă **n = 10** atunci, după apel, subprogramul va returna valoarea **6.**
(10 puncte)

3. Fișierul **bac.txt** conține un șir format din cel puțin 2 și cel mult 10^9 numere naturale, fiecare număr având cel mult 9 cifre. Să se determine o **secvență de lungime maximă**, din șirul aflat în fișier, **cu proprietatea că ultima cifră a primului număr din secvență este egală cu ultima cifră a ultimului număr din secvență**. Scrieți un program C++ care să determine și să afișeze pe ecran lungimea acestei secvențe și numărul de ordine în fișier al primului număr din secvență, utilizând un algoritm eficient din punct de vedere al timpului de execuție. Dacă în șirul de numere din fișier nu există o astfel de secvență se va afișa mesajul **Nu există.** Dacă șirul conține mai multe astfel de secvențe, se va utiliza ultima secvență din fișier. Se consideră că primul număr din fișier are numărul de ordine egal cu 1.
Exemplu: Dacă fișierul **bac.txt** conține numerele 2 37 27 24 315 144 7 85 214 atunci se afișează pe ecran 6 4.
a) Descrieți în limbaj natural algoritmul proiectat și justificați eficiența acestuia. **(2 puncte)**
b) Scrieți programul C/C++ corespunzător algoritmului descris. **(8 puncte)**

Teza 18

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Stabiliți care dintre următoarele expresii C++ are valoarea **1** dacă și numai dacă numărul întreg n este impar și divizibil cu 5.
a) !(n%2==0 && n%5==0)
b) !(n%2==0) && n%5!=0
c) !(n%2==0) && (n%5==0)
d) (n%2==1) || (n%5==0)

2. Ce valoare are funcția următoare la apelul f(5234)?
```
int f(int x)
{ if (x < 9)
   return x*x;
  else
   if(x%2==0)
        return x%10+f(x/10);
   else
        return (x%10)* f(x/10);
}
```
a) 14 **b)** 85 **c)** 120 **d)** 26

3. Un arbore cu 7 noduri, numerotate de la 1 la 7 este reprezentat prin vectorul de „tați" (2, 0, 4, 2, 4, 7, 2). Stabiliți un nod ce poate fi ales ca rădăcină a arborelui, astfel încât numărul de niveluri sa fie maxim. Nivelul pe care este situat un nod este egal cu numărul de noduri parcurse de la rădăcină până la acesta.
a) 1 **b)** 4 **c)** 6 **d)** 2

4. Utilizând metoda backtracking, se generează toate posibilitățile de a forma buchete formate din cinci flori diferite, alese dintre florile `{lalea, narcisă, crin, crizantemă, trandafir, garoafă, frezie}`. Stabiliți câte buchete de flori se pot obține, încât fiecare buchet să nu conțină împreună `crin` și `trandafir`. Ordinea florilor alese într-un buchet nu contează.
a) 21 **b)** 24 **c)** 10 **d)** 11

5. Un graf neorientat notat G este graf complet și eulerian. Care este numărul maxim de muchii pe care îl poate avea graful G, știind că este format din cel mult 50 de noduri?
a) 1225 **b)** 1176 **c)** 1128 **d)** 1275

SUBIECTUL II (40 de puncte)

1. Se consideră declararea de mai jos, corespunzătoare unei fracții reprezentată prin două numere naturale, ce reprezintă numărătorul și numitorul fracției. Variabila x memorează datele pentru 20 de fracții. Scrieți o secvență de program C++ care determină și afișează numărul de fracții care au cel mai mic numitor. **(6 puncte)**

```
struct fractie
{int numitor,numarator;
};
fractie x[20];
```

2. În secvența de instrucțiuni de mai jos variabilele **i** și **j** sunt de tip întreg, iar variabila **A** memorează un tablou bidimensional de tip întreg, cu 5 linii și 5 coloane, numerotate de la 0 la 4. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, tabloul memorat în variabila **A** să aibă elementele din figura de mai jos. **(6 puncte)**

```
for(i=0;i<5;i++)
  for(j=0;j<5;j++)
     ..............
```

```
2  2   2  2   2
2  4   6  8   10
2  6  12  20  30
2  8  20  40  70
2 10  30  70  140
```

3. Se consideră algoritmul alăturat, descris în pseudocod.
S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

```
citește a (a număr natural nenul)
m←0
t←0
┌repetă
  x←a
  d←2
  nr←0
  ┌cât timp a>1 execută
  │ ┌dacă a%d=0
  │ │ atunci
  │ │   ┌cât timp a%d=0 execută
  │ │   │   a=[a/d]
  │ │   └
  │ │   nr ←nr+1
  │ └
  │ d ←d+1
  └
  ┌dacă nr>=m
  │ atunci
  │   m←nr
  │   t←x
  └
  citește a
└până când a=0
  scrie t
```

a) Scrieți ce se va afișa dacă se citesc în această ordine numerele 4, 7, 12, 35, 0. **(6 puncte)**
b) Scrieți un set de date de intrare care să determine afișarea unui număr prim. **(6 puncte)**
c) Scrieți programul C++ corespunzător algoritmului dat. **(10 puncte)**
d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască structura „repetă... până când" cu un alt tip de structură repetitivă. **(6 puncte)**

SUBIECTUL III (30 de puncte)

1. Se citește de la tastatură un cuvânt **c**, format din cel mult 20 de litere mici și un text cu cel mult 200 de caractere format din mai multe propozitii. O propoziție conține cuvinte separate prin spațiu și se încheie cu punct. Scrieți un program C++ care va afișa pe ecran o propoziție din text care conține cuvântul c de cele mai multe ori. De exemplu, dacă se citește cuvântul **avion** și textul **Din avion vezi un cer albastru și un alt avion. Pe aeroport aterizează un avion.** Se va afișa propoziția **Din avion vezi un cer albastru și un alt avion.** **(10 puncte)**

2. Subprogramul **perechi** primește prin parametrul **n** un număr natural (**2≤n≤50**) și prin parametrul **v** primește un tablou unidimensional, format din **n** numere naturale distincte, fiecare număr având cel mult **9** cifre. Subprogramul returnează numărul de perechi distincte formate din elemente din tabloul v, cu proprietatea că un număr este oglindit pentru celălalt număr. Scrieți în limbajul C++ definiția completă a subprogramului perechi.

Exemplu: Dacă **n = 7** și **v = (41, 324, 14, 423, 82, 6, 28)** atunci, după apel, subprogramul va returna valoarea **3**, deoarece vectorul are 3 perechi de numere ce verifică proprietatea dată: (41,14); (324, 423); (82, 28). **(10 puncte)**

3. Fișierul **bac.txt** conține un șir format din cel mult 10 000 numere naturale separate prin spațiu, fiecare număr având cel mult 5 cifre. Prin alegerea cifrelor distincte din șir, care au modulul diferenței cifrelor egal cu 2 se pot obține mai multe numere. Scrieți un program C++ eficient ca timp de execuție și spațiu de memorie utilizat, care determină și afișează pe ecran cel mai mare număr format din toate cifrele numerelor din fișier, cu proprietatea că modulul diferenței cifrelor distincte este egal cu 2 sau afișează mesajul `Nu există`, dacă din șir nu se poate obține un asemenea număr.

Exemplul 1: Dacă fișierul **bac.txt** conține numerele 1012 3367 9837 369 96 183, se pot obține mai multe numere care au modulul diferenței cifrelor distincte egal cu 2 : 20, 31, 311, 97, 79, 886, 9977 etc. Numărul care se va afișa este 33333111.

Exemplul 2: Dacă fișierul **bac.txt** conține numerele 27 388 723 82, se va afișa mesajul Nu există.

a) Descrieți în limbaj natural algoritmul proiectat și justificați eficiența acestuia. **(2 puncte)**
b) Scrieți programul C/C++ corespunzător algoritmului descris. **(8 puncte)**

Teza 19

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Stabiliți care dintre următoarele expresii C++ are valoarea 1, dacă și numai dacă numerele naturale nenule x și y au aceeași paritate (ambele sunt numere pare sau ambele sunt numere impare).
a) x+y %2==0 **b)**(x+y) %2==0 **c)** (x*y) %2==0 **d)** (x*y) %2==1

2. Se consideră funcțiile f1 și f2 definite mai jos. Stabiliți ce valoare are apelul f2(10, 23).

```
int f1(int n,int d)
{ if(d<=n/2)
if(n%d==0)
return d;
else
return f1(n,d+1);
else
return 0;
}
```

```
int f2(int a,int b)
{ if(a<b)
   if(f1(a,2)>0)
     return f2(a+1,b);
   else
     return 1+f2(a+1,b);
   else
     return 0;
}
```

a) 4 **b)** 5 **c)** 9 **d)** 10

3. Se consideră un arbore cu 5 noduri, notate de la 1 la 5, reprezentat prin matricea de adiacență de mai jos. Numim înălțime a unui arbore cu rădăcină, numărul de muchii al celui mai lung lanț elementar care are una dintre extremități în rădăcina arborelui. Care este nodul ales ca rădăcină astfel încât înălțimea arborelui să fie minimă?

```
0 1 1 0 0
1 0 0 1 0
1 0 0 0 1
0 1 0 0 0
0 0 1 0 0
```

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

4. Prin metoda backtracking, se generează în ordine lexicografică toate șirurile formate din 5 litere mici distincte, cu proprietatea că primul și ultimul caracter sunt vocale. Primele trei șiruri obținute sunt în ordine: abcde, abcdi, abcdo. Știind că în alfabetul englez există 26 de litere mici, stabiliți câte șiruri care încep cu litera a se pot forma.
a) 16192 **b)** 3120 **c)** 358800 **d)** 48756

5. Fie un graf orientat G cu 6 noduri, reprezentat prin următoarele liste de adiacență: 1: (2, 3), 2: (6), 3: (2), 4: (5), 5:(4), 6: (4). Care dintre următoarele afirmații este falsă?
a) Graful G nu este tare conex.
b) Graful G are 5 componente tare conexe.
c) Graful G are 4 componente tare conexe.
d) Graful G devine tare conex dacă se adaugă cel puțin două arce noi.

SUBIECTUL II (40 de puncte)

1. În declararea următoare, variabila **c** memorează pentru fiecare 20 de cuvinte: cuvântul prin câmpul **cuv** și frecvența sa într-un text prin câmpul **frecv.** Scrieți o secvență de program C++ care determină și afișează lungimea maximă a cuvintelor memorate care au frecvența mai mare decât 2. **(6 puncte)**

```
struct cuvant
 { char cuv[30];int frecv;
 };
cuvant c[20];
```

2. În secvența de instrucțiuni de mai jos variabilele **i** și **j** sunt de tip întreg, iar variabila **A** memorează un tablou bidimensional de tip caracter, cu 4 linii și 4 coloane, numerotate de la 0 la 3. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, tabloul memorat în variabila **A** să aibă elementele din figura de mai jos. **(6 puncte)**

```
for(i=0;i<4;i++)
  for(j=0;j<4;j++)
     ..............
```

```
a a b b
A A B B
a a b b
A A B B
```

3. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

```
citește a,b (a≤ b, a, b numere naturale)
k←0; q←0
pentru x←1, a execută
    pentru y←1, b execută
        dacă y%x=0
        atunci
            k←k+y
        altfel
            q←q+y
scrie k," ",q
```

a) Scrieți valoarea care se va afișa pentru a = 3 și b = 7. **(6 puncte)**

b) Scrieți un set de valori pentru a și b astfel încât rezultatul afișat să fie k = 8 și q = 4. **(6 puncte)**

c) Scrieți programul C++ corespunzător algoritmului dat. **(10 puncte)**

d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască prima structură „pentru" cu alt tip de structură repetitivă. **(6 puncte)**

SUBIECTUL III (30 de puncte)

1. Se citesc de la tastatură două șiruri de caractere **s1** și **s2** formate din cel mult 40 de litere mici. Scrieți un program C++ care modifică în memorie cele două șiruri astfel: elimină vocalele din șirul **s1** și le inserează în șirul **s2** pe poziții pare, în ordine începând de la primul caracter. Să se afișeze pe ecran șirurile modificate.
   Exemplu: Pentru șirurile `elicopter` și `cartile`, șirul **s1** devine `lcptr`, iar **s2** devine `eciaoretile`. **(10 puncte)**

2. Subprogramul **factor** are patru parametri: **n** prin care primește un număr natural (**1≤n≤50**), **a** prin care primește un tablou unidimensional format din n numere naturale nenule care au cel mult 3 cifre, **fp** prin care furnizează cel mai mic factor prim comun tuturor elementelor din tabloul a și **p** prin care furnizează puterea factorului fp pentru care numărul fp^p este divizor pentru fiecare element din tabloul a, cu p>0. Dacă elementele tabloului a nu au un factor prim comun atunci fp are valoarea –1 și p are valoarea 0. Scrieți în limbajul C++ definiția completă a subprogramului factor.
   Exemplu: Pentru n = 4 și tabloul `a=(60,36,18,24)`, subprogramul factor va avea fp = 2, p = 1, iar pentru n = 4 și tabloul `a=(22,60,21,25)`, fp = `-1` și p = 0. **(10 puncte)**

3. Scrieți un program C++, eficient ca timp de execuție și spațiu de memorie utilizat, care afișează în fișierul **bac.txt**, în ordine strict crescătoare, pe câte o linie, toate palindroamele formate din 6 cifre impare. Primele 3 numere afișate în fișier sunt: 111111,113311,115511.

   a) Descrieți în limbaj natural algoritmul proiectat și justificați eficiența acestuia. **(2 puncte)**
   b) Scrieți programul C/C++ corespunzător algoritmului descris. **(8 puncte)**

Teza 20

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Stabiliți care dintre următoarele expresii C++ are valoarea 1 dacă și numai dacă numerele naturale nenule x și y sunt ambele divizibile cu 5.
   a) (x+y)%5==0 **b)** (x*y)%5==0 **c)** (x/5+y/5)== (x+y)/5 **d)** x+y%5==0

2. Se consideră funcția de mai jos.

```
void f(int n)
{ int i;
  if(n<5)
    cout<<"*";
   else
    if(n<10)
     cout<<"**";
    else
      for(i=1;i<=n/5;i++)
      {f(n/5);
       cout<<"*";
       }
}
```

   Alegeți 3 valori distincte ale numărului n, pentru care la apelul f(n) se vor afișa 18 caractere asterisc.
   a) 10, 12, 14 **b)** 32, 33, 34 **c)** 24, 25, 26 **d)** 20, 21, 22

3. Se consideră un arbore cu 6 noduri, reprezentat prin vectorul de „tați" t = (3, 3, 0, 3, 2, 2). Numim înălțime a unui arbore cu rădăcină, numărul de muchii al celui mai lung lanț elementar care are una dintre extremități în rădăcina arborelui. Ce înălțime are arborele dat?
   a) 3 **b)** 4 **c)** 1 **d)** 2

4. Prin metoda backtracking, se generează în ordine lexicografică toate modalitățile de scriere a unui număr natural ca sumă de numere naturale nenule și pare. Pentru numărul 6 se obțin soluțiile: 2 + 2 + 2, 2 + 4, 6. A patra soluție obținută prin același algoritm pentru numărul 10 este:
   a) 2+2+6 **b)** 2+8 **c)** 2+4+4 **d)** 2+2+2+4

5. Fie un graf neorientat G cu 6 noduri, notate de la 1 la 6, reprezentat prin următoarele liste de adiacență: 1: (2, 3), 2: (1, 3), 3: (1, 2, 4), 4: (3), 5: (6), 6: (5). Care dintre următoarele afirmații este adevărată pentru graful G?
   a) Graful este eulerian.
   b) Graful nu conține cicluri.
   c) Graful este conex.
   d) Graful nu are noduri izolate.

SUBIECTUL II (40 de puncte)

1. În declararea următoare, variabila **e** memorează pentru fiecare dintre 30 de elevi: numele elevului prin câmpul `nume`, prima limbă străină studiată prin câmpul `l1` și a doua limbă străină studiată prin câmpul `l2`. Scrieți o secvență de program C++ care determină și afișează numărul de elevi care studiază limba engleză ca prima sau a doua limbă străină. **(6 puncte)**

```
struct elev{
char l1[30],l2[30];char nume[40];};
elev e[30];
```

2. În secvența de instrucțiuni de mai jos variabilele **i** și **j** sunt de tip întreg, iar variabila **A** memorează un tablou bidimensional de tip întreg, cu 4 linii și 4 coloane, numerotate de la 0 la 3. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, tabloul memorat în variabila A să aibă elementele din figura de mai jos. **(6 puncte)**

```
for(i=0;i<4;i++)
  for(j=0;j<4;j++)
     ..............
```

```
1 2 3 7
1 1 7 4
1 7 1 4
7 2 3 1
```

3. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

```
citește a (a număr natural nenul)
m ←0
pentru d ←a, 2, –1 execută
    dacă a%d=0
        dacă d%2=1
          atunci
            m ←d
          altfel
            m ←2*d
scrie m
```

a) Scrieți ce se va afișa dacă se citește numărul 35. **(6 puncte)**
b) Scrieți un set de date de intrare care să determine afișarea numărului citit. **(6 puncte)**
c) Scrieți programul C++ corespunzător algoritmului dat. **(10 puncte)**
d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască structura „pentru” cu alt tip de structură repetitivă. **(6 puncte)**

SUBIECTUL III (30 de puncte)

1. Se citește de la tastatură un șir format din cel mult 20 de litere și cifre. Scrieți un program C++ care va determina și va afișa pe ecran produsul numerelor din șir, știind că un număr are cel mult 3 cifre și în șir există cel puțin cinci litere.
   Exemplu: Dacă se citește șirul **abc123DE2c42e5** se va afișa numărul 51660 care reprezintă produsul numerelor 123, 2, 42 și 5. **(10 puncte)**

2. Subprogramul **produs** are doi parametri: **n** prin care primește un număr natural (**1≤n≤100**) și **x** prin care primește un tablou unidimensional format din n numere întregi și returnează produsul maxim obținut din două elemente situate pe poziții distincte ale tabloului.
   Exemplu: Pentru n = 6 și tabloul a = (– 60, 36, – 8, – 2, 10, 5), subprogramul va returna 480. Scrieți în limbajul C++ definiția completă a subprogramului produs. **(10 puncte)**

3. Considerând un număr natural **x** ce conține cel puțin trei cifre, după ce se elimină prima și ultima cifră se obține un alt număr denumit **număr interior** al numărului x. Fișierul **bac.in** conține un șir având cel mult un milion de numere naturale care au cel puțin trei cifre și cel mult 5 cifre, separate prin câte un spațiu. Scrieți un program C++ care afișează pe ecran în ordine descrescătoare, separate prin spațiu, numerele interioare ale numerelor din fișier pentru care prima și ultima lor cifră sunt egale sau afișează mesajul Nu există, dacă fișierul nu conține niciun număr având prima și ultima cifră egale. Utilizați un algoritm eficient ca timp de execuție.
   Exemplu: Dacă fișierul conține șirul de numere 1151 234 2322 212 514 23122 atunci se vor afișa numerele 312 32 15 1.
   a) Descrieți în limbaj natural algoritmul proiectat și justificați eficiența acestuia. **(2 puncte)**
   b) Scrieți programul C/C++ corespunzător algoritmului descris. **(8 puncte)**

Teza 21

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Se consideră o secvență de program C++, cu variabilele întregi i, j, n și s.

```
s=0;i=1;
do
{if(i+s<=n)
   {s=s+i;
    for(j=1;j<=i;j++)
    cout<<j<<" ";
    i++;
   }
else
s=n;
}
while(s<n);
```

Ce valoare are numărul n astfel încât la execuția acestei secvențe de program șirul numerelor afișate să conțină numărul 1 de patru ori?
a) 27 **b)** 16 **c)** 11 **d)** 4

2. Ce valoare are funcția următoare la apelul f(3,14)?

```
int f(int n,int m)
{if(n<m)
  if(n%2==0)
    return n+f(n+3,m);
  else
   return n*f(n+3,m);
else
return 1;
}
```

a) 162 **b)** 97 **c)** 369 **d)** 45

3. Se consideră un graf neorientat G cu 26 de noduri, notate cu toate literele mici din alfabetul englez. În graful G, oricare două noduri notate cu vocale sunt adiacente și oricare două noduri notate cu consoane sunt adiacente. Câte muchii are graful G?
a) 325 **b)** 256 **c)** 64 **d)** 220

4. Utilizând metoda backtracking, se generează în ordine crescătoare toate numerele naturale formate din patru cifre distincte și nenule cu proprietatea că au suma cifrelor un număr par. Primele trei numere obținute sunt: 1234, 1236, 1238. Ultimele trei numere generate sunt:
a) 9874,9875,9876 **c)** 9894,9896,9898
b) 8972,8974,8976 **d)** 9872,9874,9876

5. Se consideră un arbore cu rădăcină având **n** noduri. Fiecare nod de pe nivelul i are 2*i + 1 fii, cu excepția ultimului nod de pe ultimul nivel, obținut prin parcurgerea nivelului de la stânga la dreapta, care poate avea mai puțini fii. Știind că rădăcina se află pe nivelul 0, determinați valoarea lui **n** astfel încât pe nivelul 3 din arbore să se afle 11 noduri.
a) 22 **b)** 16 **c)** 25 **d)** 32

SUBIECTUL II (40 de puncte)

1. În declararea următoare, variabila **c** memorează pentru fiecare dintre cei 200 de candidați de la examenul de Bacalaureat: numele elevului prin câmpul `nume`, denumirea probei E_d alese prin câmpul `probaE_d` și nota obținută la această probă prin câmpul `nota`. Scrieți o secvență de program C++ care afișează numele elevilor care au obținut notă mai mare decât 9 la proba Informatica. **(6 puncte)**

```
struct candidat
{char nume[30],probaE_d[30];int nota;};
 candidat c[200];
```

2. Completați punctele de suspensie din secvența de instrucțiuni de mai jos, astfel încât la execuția secvenței să se afișeze pe ecran **info programinfo**. **(6 puncte)**

```
char s[30],t[20],x[20];
strcpy(s,"informatica");strcpy(t,"programare");
.......................
cout<<x<<" "<<t;
```

3. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

```
citește n (n număr natural nenul)
s ←0; d←2
pentru i ←1, n execută
 citește a
 dacă a%d=0 atunci
   s←s+d
 altfel
   s←s+1
   d←d+1
scrie s
```

a) Scrieți ce se va afișa dacă se citesc numerele 7 22 8 21 15 7 28 3. **(6 puncte)**
b) Scrieți două seturi distincte de date de intrare care să determine afișarea numărului 10. **(6 puncte)**
c) Scrieți programul C++ corespunzător algoritmului dat. **(10 puncte)**
d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască structura „pentru” cu alt tip de structură repetitivă. **(6 puncte)**

SUBIECTUL III (30 de puncte)

1. Scrieți un program C++ care citește de la tastatură un număr natural **n** și elementele, numere întregi, ale unui tablou bidimensional A, având **n** linii și **n** coloane ($2 \le n \le 40$), determină valorile minime din coloanele tabloului A, care au elemente dispuse în ordine strict crescătoare și afișează pe ecran media lor aritmetică sau afișează mesajul **Nu există coloane ordonate strict crescător**, dacă tabloul A nu are nicio coloană cu această proprietate.
Exemplu: Pentru n = 3 și tabloul se va afișa media aritmetică a valorilor minime 2 și 7 care este egală cu 4.5. **(10 puncte)**
2. Subprogramul **termeni** are 3 parametri:
   – parametrul **n,** prin care primește un număr natural, având cel mult 8 cifre, n>1;
   – parametrul **m,** prin care furnizează cel mai apropiat termen din șirul lui Fibonacci, mai mic decât n;
   – parametrul **t,** prin care furnizează cel mai apropiat termen din șirul lui Fibonacci, mai mare decât n.
   Scrieți în limbajul C++ definiția completă a subprogramului **termeni.**
   Exemplu: Dacă n = 30 se obțin: m = 21, t = 34. **(10 puncte)**

3. Un șir de numere $x_1, x_2, \ldots x_k, \ldots x_m$ conține o **secvență-munte** dacă are un element x_k $(1<k<m)$ și există i și p $(1 \le i < k < p \le m)$ cu proprietatea: $x_i < x_{i+1} < \ldots < x_k$ și $x_k > x_{k+1} > \ldots > x_p$. Elementul x_k este denumit **vârf** și se notează cu **V**. Fișierul **bac.txt** conține pe prima linie un număr natural **n** $(4 \le n \le 10000)$ și pe doua linie un șir de n numere naturale nenule, având cel mult trei cifre. Scrieți un program C++, eficient ca timp de execuție și spațiu de memorie utilizat, care afișează pe ecran în ordine crescătoare, toate valorile V din toate secvențele-munte distincte ce aparțin șirului de numere din fișier sau afișează mesajul **Nu există**, dacă nu există nicio secvență-munte în șir.
Exemplu: Pentru n = 14, șirul de numere 9, 8, **30**, 11, 14, 20, **125**, 24, 19, 34, **89**, 80, 56, 170 conține trei secvențe-munte cu vârfurile 30,125 și 89 și se va afișa pe ecran 30, 89, 125.
a) Descrieți în limbaj natural algoritmul proiectat și justificați eficiența acestuia. **(2 puncte)**
b) Scrieți programul C/C++ corespunzător algoritmului descris. **(8 puncte)**

Teza 22

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Stabiliți ce valori au variabilele întregi a, b și c, astfel încât expresia `a/5+12.5 -b/3*c` să aibă valoarea `2.5` în limbajul C++.
   a) `a=16, b=14, c=4`
   b) `a=2, b=10, c=3`
   c) `a=11, b=14, c=3`
   d) `a=2, b=16, c=3`

2. Se consideră un vector v = (6, –2, –10, 3, –15, –1, –8). Știind ca indicii elementelor din vector încep de la 0 stabiliți ce valoare are funcția următoare la apelul f(7).

```
int f(int n){
if(n>0)
 if((n-1)%2==1&&v[n-1]<0)
  return v[n-1]+f(n-1);
 else
 return f(n-1);
else
 return 0;
}
```

   a) –11 **b)** –25 **c)** –3 **d)** 0

3. Se consideră un graf neorientat reprezentat prin matricea de adiacență alăturată și operațiile **AD (x, y)**: adăugarea în graf a muchiei de la nodul x la nodul y și **EL (x, y)**: eliminarea din graf a muchiei de la nodul x la nodul y. Stabiliți care dintre secvențele de operații următoare aplicate grafului îl transformă în arbore.

0	1	0	1	0	0
1	0	0	1	0	0
0	0	0	0	0	0
1	1	0	0	0	0
0	0	0	0	0	1
0	0	0	0	1	0

   a) AD(2,3); AD(3,6); EL(5,6)
   b) EL(1,2); EL(1,4); AD(3,6)
   c) EL(2,4); AD(2,3); AD(3,5)
   d) AD(2,5); AD(2,6); AD(2,3)

4. Se generează prin metoda backtracking șiruri de 4 caractere formate din caracterele `'+'` și `'-'` astfel încât orice șir generat nu conține două caractere `'-'` alăturate. Primele cinci

soluții obținute sunt `++++, +++-, ++-+, +-++, +-+-`. Care sunt ultimele două șiruri obținute prin această generare?

a) `---+, --+-` **b)** `-+++, -++-` **c)** `-++-, -+-+` **d)** `---+, ++-+`

5. Un graf orientat format din 5 noduri este reprezentat prin listele de adiacență următoare: 1: (3, 5); 2: listă vidă; 3: (2); 4: (5); 5: listă vidă.
Care este numărul minim de arce ce trebuie adăugate grafului, astfel încât graful să fie tare conex?

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

SUBIECTUL II (40 de puncte)

1. Se consideră declarările de mai jos, în care tabloul **c** memorează punctajele obținute de 300 de participanți la un concurs sportiv, format din mai multe probe. Tabloul c este ordonat după câmpul **proba**. La fiecare probă participă mai mulți concurenți.

```
struct concurs{
char nume_concurent[25],proba[20];
int punctaj;}c[300];
```

Scrieți o secvență de instrucțiuni C++ care să afișeze pe ecran pentru fiecare probă din concurs, denumirea probei și media aritmetică a punctajelor obținute de concurenții participanți la proba respectivă. **(6 puncte)**

2. În secvența de instrucțiuni de mai jos, variabilele **i** și **j** sunt de tip întreg, iar variabila **A** memorează un tablou bidimensional de tip char, cu 5 linii și 5 coloane, numerotate de la 0 la 4. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, tabloul memorat în variabila A să aibă elementele din figura de mai jos. **(6 puncte)**

```
for(i=0;i<5;i++)
 for(j=0;j<5;j++)
 ..............
```

```
a b c d e
b a b c d
c b a b c
d c b a b
e d c b a
```


3. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

a) Scrieți ce se va afișa dacă se citesc numerele 135, 23, 122, 76, 44, 124, 0. **(6 puncte)**
b) Scrieți un set de date de intrare format din cel puțin patru numere care să determine afișarea numărului 0. **(6 puncte)**
c) Scrieți programul C++ corespunzător algoritmului dat. **(10 puncte)**
d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască structura „cât timp" cu alt tip de structură repetitivă. **(6 puncte)**

```
citește x (x număr natural nenul, x>9)
k ←0
┌cât timp x≠0 execută
 citește y (y număr natural)
   ┌dacă x/10%10= y%10 și y≠0
   atunci
    k ←k+1
   ■
 x ←y
■
 scrie k
```

SUBIECTUL III (30 de puncte)

1. Se citește de la tastatură un text cu cel mult 500 de caractere (litere și spații), format din cuvinte separate între ele printr-un spațiu. Se citește un cuvânt **c**, format din cel mult 10 litere mici. Să se elimine din text toate aparițiile cuvântului **c** care sunt precedate de un cuvânt care începe cu o vocală. Să se afișeze textul dat, după modificarea sa în memorie sau să se afișeze mesajul **Textul nu s-a modificat**, în cazul în care nu s-a eliminat niciun cuvânt.
Exemplu: Dacă se citește textul **Firma implementează proiecte de design interior proiecte de consolidare și proiecte de amenajări** și se citește cuvântul **proiecte**, se va afișa textul modificat: **Firma implementează de design interior de consolidare și proiecte de amenajări.** **(10 puncte)**

2. Subprogramul **prime** primește prin parametrul **n** un număr natural ($2 \le n \le 50$) și prin parametrul **a** primește un tablou unidimensional, format din **n** numere naturale nenule și distincte, fiecare număr având cel mult **9** cifre. Subprogramul returnează numărul de perechi distincte formate din elemente din tabloul v, prime între ele. Două numere naturale sunt prime între ele dacă au un singur divizor comun: numărul 1. Scrieți în limbajul C++ definiția completă a subprogramului prime.
Exemplu: Dacă **n = 6** și **a = (42, 3, 14, 43, 8, 6)** atunci, după apel, subprogramul va returna valoarea **7**, deoarece vectorul are 7 perechi de numere ce verifică proprietatea dată: (42, 43); (3, 14); (3, 43); (3, 8); (14, 43); (43, 8); (43, 6). **(10 puncte)**

3. Fișierul **bac.txt** conține pe prima linie un număr natural **n**, care este multiplu de 7 ($7 \le n \le 70000$) și pe a doua linie un șir de n numere întregi nenule, având cel mult trei cifre.

Șirul de numere se împarte în secvențe de câte 7 numere denumite **benzi** și pentru fiecare bandă se determină cele mai mici două numere distincte **min1** și **min2**. Scrieți un program C++, eficient ca timp de execuție și spațiu de memorie utilizat, care afișează pe ecran, pentru fiecare dintre benzi, în ordine crescătoare cele două valori **min1** și **min2** sau afișează numărul 0, dacă banda nu conține două valori distincte **min1** și **min2.**

Exemplu: Pentru **n = 21**, șirul de numere –3, 10, 1, 2, –5, 6, –5, 1, 1, 1, 1, 1, 1, 1, 10, 9, 1, 2, 3, 6, 7 se va afișa –5 –3 0 1 2.

a) Descrieți în limbaj natural algoritmul proiectat și justificați eficiența acestuia. **(2 puncte)**

b) Scrieți programul C/C++ corespunzător algoritmului descris. **(8 puncte)**

Teza 23

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Stabiliți ce valori au variabilele întregi **n** și **x**, astfel încât la execuția secvenței de program următoare să se afișeze patru caractere * urmate de șase caractere #.

```
int i=1,n,x;
while(i*i<=n)
{cout<<'*'; i++;}
while(i<=n+x)
{cout<<'#';i++;}
```

a) n = 18, x = 2 **b)** n = 5, x = 6 **c)** n = 16, x = –6 **d)** n = 24, x = 5

2. Ce se va afișa la apelul **sir('f')** al funcției următoare?

```
void sir (char c){
char x;
if(c>'a')
    {x=c+1;
     cout<<x;
     sir(c-1);
     cout<<c;
}
}
```

a) gffeeddccb **b)** fedcbgfedc **c)** efcdgbefcd **d)** gfedcbcdef

3. Un arbore cu 48 noduri are următoarele proprietăți:
– rădăcina se află pe nivelul 0;
– orice nivel notat cu un număr par, cu excepția nivelului 0, are 10 noduri;
– orice nivel notat cu un număr impar are 9 noduri.

Numărul minim de frunze ale arborelui este:
a) 2 **b)** 4 **c)** 11 **d)** 16

4. Sorin are de cultivat în grădină patru tipuri de plante din cinci categorii de plante ornamentale pe care le are la dispoziție: azalee, magnolie, liliac, margarete și anemone. El dorește să nu cultive magnolia lângă liliac. Utilizând metoda backtracking, primele trei soluții generate sunt: (azalee, magnolie, margarete, anemone), (azalee, magnolie, anemone, margarete), (azalee, liliac, margarete, anemone). Care este a șasea soluție obținută?
   a) (azalee, margarete, liliac, magnolie)
   b) (azalee, anemone, liliac, margarete)
   c) (azalee, margarete, anemone, magnolie)
   d) (azalee, margarete, magnolie, anemone)

5. Se consideră un graf neorientat având 6 noduri, notate de la 1 la 6 și mulțimea muchiilor U={[1, 2], [1, 6], [2, 3], [2, 4], [2, 5], [2, 6], [3, 4], [3, 5], [4, 5], [5, 6]}. Prin eliminarea unui număr minim de muchii, graful devine eulerian. Care sunt muchiile ce trebuie eliminate?
   a) [2, 4] și [2, 5]
   b) [2, 5]
   c) [2, 6] și [4, 3]
   d) [2, 3] și [2, 6]

SUBIECTUL II (40 de puncte)

1. Se consideră declarările de mai jos, în care variabila **o** memorează date despre 200 de orașe: denumirea orașului, a țării căruia îi aparține orașul, numărul de locuitori și suprafața orașului (exprimată în număr de km²).

```
struct oras{
char tara[30],nume_oras[20];
int nr_locuitori_oras; float suprafata_oras;}o[200];
```

Știind că densitatea populației reprezintă numărul de persoane care locuiesc pe un km², scrieți o secvență de instrucțiuni în limbajul C++ care să afișeze numele orașelor și țărilor lor, pentru toate orașele care au o densitate mai mare de 15 locuitori/km². **(6 puncte)**

2. Ce se va afișa după execuția secvenței de instrucțiuni următoare?

```
char s[20],t[10];
strcpy(s,"exercitii"); strcpy(t,"***");
cout<<strlen(s)+strlen(t)<<" ";
strcat(s,t);
cout<<strchr(s,'i');
```

(6 puncte)

3. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

a) Scrieți ce se va afișa dacă se citesc numerele 12, 36, 3. **(6 puncte)**
b) Scrieți un set de date de intrare, cu proprietatea b –a = 2, pentru care se afișează trei numere. **(6 puncte)**
c) Scrieți programul C++ corespunzător algoritmului dat. **(10 puncte)**
d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască structura „pentru” cu alt tip de structură repetitivă. **(6 puncte)**

```
citește a, b, k (numere naturale
nenule, a≤b, k>1)
pentru x←a,b execută
  y ←0
  t←x
    cât timp t≠0 execută
    y←y*10+t%10
    t←[t/10]
    dacă y%k=0
      atunci
        scrie x," "
```

SUBIECTUL III (30 de puncte)

1. Scrieți un program C++ care citește de la tastatură un număr natural **n** și elementele, numere naturale, ale unui tablou bidimensional A, având **n** linii și **n** coloane ($2 \le n \le 50$) și înlocuiește toate elementele care sunt numere pare, formate din cel mult 3 cifre, cu cel mai mare element situat pe diagonala principală sau diagonala secundară a matricei, apoi afișează matricea transformată.

Exemplu: Pentru n = 3 și tabloul $A = \begin{matrix} 2 & 1000 & 7 \\ 40 & 8 & 27 \\ 5 & 2 & 9 \end{matrix}$ se va afișa matricea transformată

$A = \begin{matrix} 9 & 1000 & 7 \\ 9 & 9 & 27 \\ 5 & 9 & 9 \end{matrix}$

(10 puncte)

2. Subprogramul **cifre** primește prin parametrul **n** un număr natural nenul, format din cel mult 9 cifre, elimină toate cifrele pare ale numărului n și furnizează prin parametrul **n** numărul modificat și prin parametrul **k** un număr format din cifrele pare distincte ale numărului n, în ordinea scrierii lor. Parametrul k este egal cu –1 dacă numărul n nu are cifre pare.
Exemplu: Pentru numărul n = 12267488, se obține n = 17 și k = 2648, iar pentru n = 1331, se obține n = 1331 și k = –1. Scrieți în limbajul C++ definiția completă a subprogramului **cifre**. **(10 puncte)**

3. Fișierul **bac.in** conține un șir format din cel mult 10^9 litere ce aparțin mulțimii M = {a, A, b, B, c, C, d, D e, E}. Scrieți un program C++, eficient ca timp de execuție, ce afișează pe ecran prin litere mici, caracterele ce apar de un număr impar de ori în șirul dat, în ordinea enumerării lor în mulțimea M, fără a face distincție între litere mari și litere mici sau afișează mesajul Nu există, dacă șirul dat nu conține niciun caracter cu proprietatea menționată.
Dacă fișierul bac.txt conține caracterele `cbcCdEaAdBacCDEaacc` atunci pe ecran se va afișa `aaaaacccccccddd`. Dacă fișierul bac.txt conține caracterele `cbcBaa`, pe ecran se va afișa mesajul Nu există.
a) Descrieți în limbaj natural algoritmul proiectat și justificați eficiența acestuia. **(2 puncte)**
b) Scrieți programul C/C++ corespunzător algoritmului descris. **(8 puncte)**

Teza 24

SUBIECTUL I

(20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Se consideră două numere întregi x și y și expresia `e=(x+10)>y?x:y`. Pentru ce valori ale lui x și y, expresia este egală cu y?
a) x = 3, y = –5 **b)** x = 10, y = 20 **c)** x = 20, y = 10 **d)** x = –5, y = 3

2. Se consideră șirul de caractere s = "abddcdebc". Ce valoare are funcția următoare la apelul sir(s, 0)?

```
int sir(char s[],int i){
if(i<strlen(s)/2)
  if(s[i]==s[strlen(s)-i-1])
   return 1+sir(s,i+1);
  else
   return sir(s,i+1);
else
 return 0;
}
```

a) 0 **b)** 1 **c)** 2 **d)** 3

3. Se consideră un arbore ce conține **n** muchii ($5 \le n \le 10$). Nodurile arborelui sunt etichetate cu numere prime distincte, consecutive în mulțimea numerelor prime, începând de la 2. În ce interval sunt situate aceste numere prime?
a) [2,10] **b)** [2,31] **c)** [11,29] **d)** [29,79]

4. Având la dispoziție suma S egală cu 100 de lei și șirul de bancnote exprimate **în lei:** (10, 20, 30, 40), se poate exprima suma S cu bancnotele date în mai multe moduri.
Utilizând metoda backtracking, o modalitate de exprimare a sumei S are forma $x = (x_1, x_2, x_3, x_4)$, unde x_1 reprezintă numărul de bancnote de 10 lei, x_2 reprezintă numărul de bancnote de 20 lei, x_3 reprezintă numărul de bancnote de 30 lei și x_4 reprezintă numărul de bancnote de 40 lei.
Exemplu: Soluția (2, 2, 0, 1) exprimă suma 100 = 2*10 + 2*20 + 0*30 + 1*40. Dacă primele cinci soluții obținute sunt: (10, 0, 0, 0), (8, 1, 0, 0), (7, 0, 1, 0), (6, 2, 0, 0), (6, 0, 0, 1), atunci următoarele două soluții generate în ordine imediat după soluția (4, 3, 0, 0) sunt:

a) (4, 1, 0, 1) și (2, 4, 0, 0)
b) (3, 0, 1, 1) și (2, 4, 0, 0)
c) (4, 1, 0, 1) și (1, 1, 1, 1)
d) (4, 1, 0, 1) și (3, 0, 1, 1)

5. Un graf neorientat conține 5 noduri, notate de la 1 la 5. Oricare două noduri notate cu numere prime între ele sunt adiacente. Care dintre următoarele afirmații nu este adevărată?
   a) Graful este conex.
   b) Graful este complet.
   c) Graful este hamiltonian.
   d) Graful nu este eulerian.

SUBIECTUL II (40 de puncte)

1. Se consideră declarările de mai jos, în care tabloul **f** memorează date despre filmele difuzate la un cinematograf. Pentru fiecare film se memorează: **numele filmului**, **genul** (istoric, actiune, comedie, SF etc.), data difuzării filmului denumită **difuzare** și suma încasată din vânzarea biletelor la data difuzării denumită **suma**.

```
struct data{
int zi,luna,an;};
struct film{
char nume_film[40],gen_film[20];
data difuzare;
int suma;}f[1000];
```

Scrieți o secvență de instrucțiuni C++ care să determine și să afișeze pe ecran suma totală încasată de cinematograf, pentru toate filmele ce aparțin genurilor **comedie** și **istoric**, difuzate în luna martie, anul 2018. **(6 puncte)**

2. În secvența de instrucțiuni de mai jos variabilele **i** și **j** sunt de tip întreg, iar variabila **A** memorează un tablou bidimensional de tip întreg, cu 5 linii și 5 coloane, numerotate de la 0 la 4. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, tabloul memorat în variabila A să aibă elementele din figura de mai jos. **(6 puncte)**

```
for(i=0;i<5;i++)
  for(j=0;j<5;j++)
     ..............
```

```
 1  2  3  4 -2
 3  5  7  2 -2
 8 12  9  0 -2
20 21  9 -2 -2
41 30  7 -4 -2
```


3. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

a) Scrieți ce se va afișa dacă se citesc numerele 40 16 10 2880 18 80838 22 19818 0. **(6 puncte)**
b) Scrieți un set de date de intrare care să determine afișarea numărului 0. **(6 puncte)**
c) Scrieți programul C++ corespunzător algoritmului dat. **(10 puncte)**
d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască prima structură repetitivă cu test final din algoritm cu o structură repetitivă cu test inițial. **(6 puncte)**

```
citește a, b (a, b numere naturale
nenule, a>b)
n ←0
┌repetă
 citește x (x număr natural nenul)
 t←0
 y←x
 ┌repetă
 │  ┌dacă  x%10=a%b%10
 │  │   atunci
 │  │      t ←t+1
 │  └
 │   x←[x/10]
 └până când x=0

 ┌dacă t≥2
 │  atunci
 │    n←n+1
 └
└până când y=0
scrie n
```

SUBIECTUL III (30 de puncte)

1. Se citește de la tastatură un text, având cel mult 400 de caractere care sunt litere mici și spații, format din mai multe cuvinte separate printr-un singur spațiu. Scrieți un program C++ care modifică în memorie textul dat astfel: literele oricărui cuvânt din text care are proprietate palindromică se transformă în litere mari. Textul modificat se va afișa pe ecran.
Exemplu: Dacă textul este: **tot ce este rar cojoc si radar** atunci se va afișa **TOT ce este RAR COJOC si RADAR.** **(10 puncte)**

2. Subprogramul **maxim** primește prin parametri **n** și **m** două numere naturale ($2 \leq n, m \leq 30$) și prin parametrul **T** un tablou bidimensional format din n linii și m coloane. Elementele tabloului sunt numere naturale. Subprogramul returnează cel mai mare număr par format numai din cifre pare, din prima linie a tabloului T care începe cu un număr impar sau returnează valoarea –1 dacă nu există un astfel de număr în tablou. Scrieți în limbajul C/C++ definiția completă a subprogramului **maxim.** **(10 puncte)**

3. Fișierul **bac.in** conține un șir format din cel mult 100 000 numere naturale separate prin spațiu, care au exact 3 cifre. Scrieți un program C++, eficient ca timp de execuție și spațiu

de memorie utilizat, care determină și afișează pe ecran **lungimea maximă a unei secvențe** din șirul aflat în fișier, cu proprietatea că toate numerele secvenței încep cu aceeași cifră și **numărul de secvențe de lungime maximă** de acest tip din fișier.
Exemplu: Dacă fișierul conține numerele 103 879 867 809 876 234 243 367 343 342 885 801 786 754 724 721, atunci se va afișa **4 2**.

a) Descrieți în limbaj natural algoritmul proiectat și justificați eficiența acestuia. **(2 puncte)**

b) Scrieți programul C/C++ corespunzător algoritmului descris. **(8 puncte)**

Teza 25

SUBIECTUL I

(20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Variabilele `x, y, a, b` memorează numere naturale. Care dintre următoarele expresii este adevarată dacă și numai dacă `[x,y]⊂[a,b]`?
a) `a<=x||b<=y` **b)** `a<=x&&y<=b` **c)** `!(a<=x&&y<=b)` **d)** `x<=a&&b<=y`

2. Se consideră graful orientat cu 8 noduri numerotate de la 1 la 8 și cu arcele: (1,3),(3,2), (2,1),(3,4),(4,7),(7,8),(8,5),(5,6),(6,4). În câte moduri putem adăuga un singur arc, astfel încât graful să devină tare conex?
a) 12 **b)** 8 **c)** 1 **d)** 15

3. La examenul de Bacalaureat, un elev trebuie să genereze prin metoda backtracking permutările mulțimii {A, B, C, D, E}, în care vocalele să fie precedate de consoane. Dacă primele trei permutări generate sunt BCDAE, BCDEA, CBDAE, la a câta permutare va fi obținut șirul DCBAE?
a) 12 **b)** 10 **c)** 11 **d)** 9

4. Un graf neorientat se numește pădure dacă toate componentele sale conexe sunt arbori. Fiind dat un graf neorientat cu 10 noduri și două componente conexe grafuri complete, una cu 21 de muchii și una cu 3 muchii, determinați numărul minim de muchii care trebuie eliminate din graf astfel încât graful să devină pădure.
a) 10 **b)** 8 **c)** 16 **d)** 17

5. Se consideră subprogramul f definit mai jos!

```
void f(int x, char c)
{   int i;
    if(x>1)
    {   cout<<"*";
        f(x-1,c+1);
        cout<<c<<"#";;
    }
    else
     cout<<c;
}
```

Ce se va afișa în urma apelului f(4,'a')?
a) *#*#*#abcd **b)** *a#*b#*c#*d **c)** ***dc#b#a# **d)** *#d*#c*#b*#a

SUBIECTUL II (40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

a) Scrieți ce se va afișa dacă pentru n se citește valoarea 5. **(6 puncte)**

b) Scrieți cea mai mare valoare care poate fi citită pentru n astfel încât să se afișeze 5 valori. **(6 puncte)**

c) Scrieți programul C/C++ corespunzător algoritmului dat. **(10 puncte)**

d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să să înlocuiți structura „cât timp" cu o structură repetitivă cu test final. **(6 puncte)**

```
citește n (n număr natural)
x←7
pentru i←1, n execută
    j←2
    cât timp j≤[x/j] și x%j0 execută
        j←j+1
    dacă j*j>x atunci
        scrie x,' '
    x←x+10
```

2. Se consideră secvența de program C++ următoare:

```
char voc[]="aeiou",s[100];
int i;
strcpy(s,"*galbbbleinn*asaa*appaarei*astfel*creuzet*");
for(i=0;i<strlen(s)-1;i++)
    {...}
cout<<s;
```

Completați punctele de suspensie cu o secvență de instrucțiuni C++, astfel încât să se afișeze șirul
`*galen*asa*apare*asel*cezet*`. **(6 puncte)**

3. Se consideră declarările de mai jos, în care tabloul **t** memorează laturile pentru **n** triunghiuri. (1≤n<100) și x≤y.

```
struct triunghi{
float l1,l2,l3;}
t[100];
float x,y;
int n,i;
```

Scrieți o secvență de instrucțiuni C++ care să afișeze pe ecran laturile triunghiurilor pentru care aria aparține intervalului [x,y], separând triunghiurile prin caracterul *. **(6 puncte)**

SUBIECTUL III (30 de puncte)

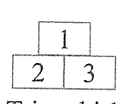
1. Scrieți un program C/C++ care citește de la tastatură un număr natural `N` (`N∈[2,10]`), construiește în memorie și afișează un tablou bidimensional cu `N` linii și `2*N-1` coloane, numere naturale din mulțimea {1, 2}, astfel încât valoarea elementelor de pe semidiagonalele care încep cu primul element din stânga sus respectiv primul element din dreapta sus să fie egale cu 1, celelalte elemente fiind egale cu 2. Elementele matricei vor fi afișate pe ecran linie cu linie separate printr-un spațiu. **(10 puncte)**

Exemplu: Dacă `N=3` atunci tabloul este:

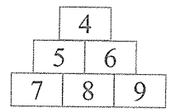
```
1 2 2 2 1
2 1 2 1 2
2 2 1 2 2
```

2. Un tablou unidimensional se numește **k-palindrom** dacă după efectuarea a `k` permutări circulare cu o poziție spre stânga acesta devine palindrom (considerăm că un vector este palindrom dacă vectorul parcurs de la stânga la dreapta coincide cu vectorul parcurs de la dreapta la stânga). Scrieți definiția completă a unui subprogram **`kpal`** cu trei parametri: `v` un tablou unidimensional cu cel mult 100 de elemente numere întregi, `n` un număr natural (`n≤100`), reprezentând numărul de elemente din vector și `k` un număr natural (`1≤k<n`), care returnează valoarea 1 dacă vectorul este k-palindrom sau valoarea 0 în caz contrar.
Exemplu: Dacă n=5, v=(2, 2, 4, 5, 4) atunci f(v,n,1) va returna valoarea 1. **(10 puncte)**

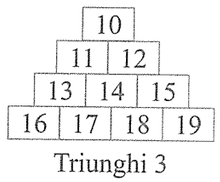
3. Cu șirul crescător al numerelor naturale de la 1 la 10^6, se construiesc triunghiuri de numere ca în imaginea de mai jos (ultimul triunghi poate fi incomplet).



Triunghi 1



Triunghi 2



Triunghi 3

Se citește de la tastatură un număr k, $k < 10^6$. Să se scrie în fișierul text **BAC.TXT** pe prima linie, separate prin câte un spațiu, valorile T L C, reprezentând: T = numărul triunghiului care conține valoarea k, L = linia pe care se află valoarea k (numerotată de sus în jos, vârf = linia 1), C = coloana acestuia (numerotată de la stânga la dreapta). Proiectați un algoritm eficient din punct de vedere al timpului de executare și al spațiului de memorie utilizat.

Exemplu: Dacă se citește de la tastatură numărul k=17 atunci în fișier se vor afișa:
`3 4 2`

a) Realizați o descriere de 3-4 rânduri a algoritmului ales justificând eficiența acestuia. **(2 puncte)**

b) Scrieți programul C/C++ corespunzător metodei descrise la punctul **a)**. **(8 puncte)**

TEZE

Specializarea Științe ale Naturii

Teza 1

SUBIECTUL I

(20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Variabilele **a, b** și **z** sunt reale, iar **a≤b.** Care dintre expresiile C/C++ următoare are valoarea **1** dacă și numai dacă valoarea variabilei **z** aparține intervalului închis determinat de valorile variabilelor **a** și **b**?
   a) `(z>a||(z>b)`
   b) `(z<a)||(z>b)`
   c) `z<a && z>b`
   d) `z>=a && z<=b`

2. Aplicând algoritmul de căutare binară pentru căutarea unei valori x în tablourile unidimensionale `(1,5,7,10,20)` și `(19,10,9,8,7,6,5)` se fac două comparații în ambele cazuri. Care dintre următoarele valori poate fi valoarea variabilei x?
   a) 5 **b)** 10 **c)** 8 **d)** 6

3. Indicați expresia care are valoarea 1 dacă și numai dacă numărul memorat în variabila întreagă x este număr natural.
   a) `floor(x)==abs(x)`
   b) `floor(x)==ceil(x)`
   c) `floor(x)-1==x`
   d) `abs(x)==x`

4. Variabilele **i** și **j** sunt de tip întreg. Indicați expresia care poate înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, să se afișeze numerele de mai jos, în această ordine.

```
for(i=1;i<=5;i++)
  {
    for(j=1;j<=5;j++)
     cout<<....<<' ';
    cout<<endl;
  }
```

```
2 3 4 0 1
3 4 0 1 2
4 0 1 2 3
0 1 2 3 4
1 2 3 4 0
```

   a) `i+j`
   b) `(i+j)%5`
   c) `(i+j-1)%5`
   d) `i+j-5`

5. Se consideră algoritmul:

```
citește x (x număr natural)
y←0
┌ cât timp x>0 execută
│      y←y+x%10
│      x←[x/100]
└■
```

Ce se va afișa dacă se citește pentru x valoarea 14360?

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

SUBIECTUL II (40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

```
citește x
(x număr natural)
┌cât timp x>9 execută
│ y←0
│ ┌ cât timp x>0 execută
│ │    y←y+x%10
│ │    x←[x/10]
│ └■
│ x←y
└■
scrie x
```

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

a) Scrieți valoarea care se va afișa pentru x = 2019. **(6 puncte)**

b) Scrieți cel mai mic număr de două cifre și cel mai mare număr de două cifre care pot fi citite pentru x astfel încât rezultatul afișat să fie 9. **(6 puncte)**

c) Scrieți programul C/C++ corespunzător algoritmului dat. **(10 puncte)**

d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se nu se utilizeze structuri repetitive. **(6 puncte)**

2. Se consideră declarările de mai jos, în care variabila `nr` memorează numărul de vârfuri ale unui poligon (număr natural din intervalul $[3, 10^2]$) și tablourile `x` și `y` memorează coordonatele vârfurilor acestuia (abscisa și ordonata) în sistemul de coordonate xOy (numere reale). Fără a utiliza variabile suplimentare scrieți o secvență de instrucțiuni C/C++ care determină și afișează pe ecran perimetrul poligonului memorat în variabila `per`. **(6 puncte)**

```
int i,nr;
float per,x[101],y[101];
```

3. În urma utilizării algoritmului de interclasare descrescătoare a două tablouri unidimensionale cu elemente distincte se obține tabloul `(20,19,10,10,9,8,7,5,5,1,1)`. Dați exemplu de două astfel de tablouri ordonate crescător astfel încât să se obțină rezultatul precizat. **(6 puncte)**

SUBIECTUL III (30 de puncte)

1. Se citește un număr natural n și se cere să se scrie suma divizorilor primi ai lui n.
Exemplu: Dacă n = 12, atunci se scrie 5, iar dacă n = 11 atunci se scrie 11. Scrieți, în pseudocod, algoritmul de rezolvare al problemei enunțate. **(10 puncte)**

2. Scrieți un program C/C++ care citește de la tastatură numărul natural n ($n \in [2,10]$, apoi cele n elemente ale unui tablou unidimensional, numere naturale din intervalul $[0, 10^2]$, și afișează pe ecran, separate printr-un spațiu, primul număr și ultimul număr memorate în tablou, apoi pe linia următoare primele două numere din tablou și ultimele două numere din tablou, ș.a.m.d. în final primele n numere din tablou și ultimele n numere din tablou ca în exemplu. **(10 puncte)**
Exemplu: Dacă `n=4` și elementele tabloului sunt `2 4 7 1` atunci pe ecran se va afișa:

```
2 1
2 4 7 1
2 4 7 4 7 1
2 4 7 1 2 4 7 1
```

3. Scrieți un program C/C++ care citește din fișierul text **BAC.TXT** un șir S cu cel mult 100 000 de numere naturale formate din cel mult trei cifre fiecare. Asupra acestui șir se aplică în mod repetat următoarea prelucrare: se elimină din șir valorile prime, iar valorile neprime se incrementează cu valoarea 1. Prelucrarea se repetă până când în șir nu mai rămâne niciun număr. Să se afișeze pe ecran de câte ori a fost aplicată această prelucrare.
Exemplu: Dacă fișierul conține numerele 12 11 16 45 34 atunci după prima prelucrare vom avea numerele 13 17 46 35; după a doua 47 36; după a treia 37; astfel pe ecran va fi afișată valoarea 4.

a) Se cere să se proiecteze un algoritm eficient din punct de vedere al timpului de executare și al spațiului de memorie utilizat și să se realizeze o descriere de 3-4 rânduri a algoritmului ales justificându-se eficiența acestuia. **(2 puncte)**

b) Scrieți programul C/C++ corespunzător metodei descrise la punctul **a)**. **(8 puncte)**

Teza 2

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Variabilele **a, b** sunt numere naturale. Care dintre expresiile C/C++ are valoarea 1 dacă și numai dacă valorile variabilelor a și b sunt numere naturale consecutive?
   a) `a-b==1`
   b) `a==1&&b==0`
   c) `a-b==1&&b-a==1`
   d) `a-b==1||b-a==1`

2. Aplicând algoritmul de căutare binară pentru căutarea unei valori x în tabloul unidimensional `(19,15,11,9,8,7,1)` se fac exact trei comparații. Care dintre următoarele valori poate fi valoarea variabilei x?
   a) `19,11` **b)** `15,7` **c)** `8,1` **d)** `19,11,8,1`

3. Variabilele n, k, nr memorează numere naturale. Variabila `nr` trebuie să memoreze numărul de numere din intervalul `[0,n]` care sunt divizibile cu `k`. Cu ce expresie trebuie completată atribuirea `nr=...?`
   a) `nr = n/k + 1`
   b) `nr = n/k`
   c) `nr = n/k + n%k`
   d) `nr = n%k + 1`

4. Variabilele **i** și **j** sunt de tip întreg. Indicați expresia care poate înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, să se afișeze numerele de mai jos, în această ordine.

```
for(i=1;i<=5;i++)
  {
    for(j=1;j<=5;j++)
     if(....)
      cout<<3<<' ';
     else
      cout<<1<<' ';
    cout<<endl;
  }
```

```
1 1 1 1 3
1 1 1 3 1
1 1 3 1 1
1 3 1 1 1
3 1 1 1 1
```

a) `i+j==5` **b)** `i==j` **c)** `i+j==6` **d)** `i%2==j%2`

5. Se consideră algoritmul:

```
x←3 (x număr natural)
pentru i ←1,100 execută
    dacă x<8 atunci x←x*2
scrie x
```

Ce se va afișa în urma execuției algoritmului?
a) 12 **b)** 32 **c)** 36 **d)** 24

SUBIECTUL II (40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

```
citește n  (n număr natural)
y←0
cât timp n > 0 execută
    citește x
    cât timp x%2=0 execută
        x←x/2
    dacă x = 1 atunci
        y←y+1
    n←n – 1
scrie y
```

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.
a) Scrieți valoarea care se va afișa dacă se citesc pe rând valorile 4, 8, 6, 16, 45. **(6 puncte)**
b) Pentru n = 4, scrieți un set de date de astfel încât rezultatul afișat să fie 0. **(6 puncte)**
c) Scrieți programul C/C++ corespunzător algoritmului dat. **(10 puncte)**
d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se utilizeze alt tip de structură repetitivă. **(6 puncte)**

2. Două puncte M și N din planul `xOy` sunt date prin coordonatele lor carteziene `XM, YM` respectiv `XN, YN`. Scrieți o expresie C++ care are valoarea 1 dacă și numai dacă cele două puncte aparțin aceleiași axe de coordonate. **(6 puncte)**

3. Se consideră tablourile unidimensionale A = (3, 8, 15, 78, 80) și B = (15, 12, 9, 7, 3). Scrieți elementele tabloului C, în ordinea în care ele apar în tablou, astfel încât acesta să fie obținut prin interclasarea descrescătoare a elementelor din A și B. **(6 puncte)**

SUBIECTUL III (30 de puncte)

1. Se citesc două numere naturale, a și b, se cere să se determine numărul de valori din intervalul închis determinat de a și b, care au exact 3 divizori.
   Exemplu: Dacă a = 3 și b = 20, atunci se scrie **2** (doar 4 și 9 au exact 3 divizori). Scrieți, în pseudocod, algoritmul de rezolvare al problemei enunțate. **(10 puncte)**

2. Scrieți un program C/C++ care citește de la tastatură numărul natural n, unde n ∈ [2,50], apoi cele n elemente ale unui tablou unidimensional, numere naturale din intervalul $[0,10^9]$, și afișează pe ecran pe linii diferite elementele distincte din tablou urmate de numărul lor de apariții. **(10 puncte)**

Exemplu: Dacă `n=5` și elementele tabloului sunt `2 4 7 4 7` atunci pe ecran se va afișa, nu neapărat în această ordine:
```
2 1
4 2
7 2
```

3. Din fișierul **`bac.txt`** se citesc `n` și `m` (numere naturale, `0<m<n<100000`) de pe prima linie, apoi `n` numere naturale cu cel mult două cifre fiecare `a1, a2, ..., an` de pe linia a doua și apoi `m` numere naturale cu cel mult două cifre fiecare `b1, b2, ..., bm` de pe linia a treia a fișierului. Să se determine câte șiruri `b` se pot obține din șirul `a` dacă se poate schimba ordinea elementelor din șirul `a`. Se va afișa pe ecran numărul numărul de șiruri obținute.
   a) Se cere să se proiecteze un algoritm eficient din punct de vedere al timpului de executare și al spațiului de memorie utilizat și să se realizeze o descriere de 3-4 rânduri a algoritmului ales justificându-se eficiența acestuia. **(2 puncte)**
   b) Să se scrie programul C/C++ ce realizează prelucrarea descrisă și afișează pe ecran un mesaj corespunzător. **(8 puncte)**

Exemplu: Pentru fișierul **`bac.txt`** cu conținutul:
```
8 3
1 6 3 1 3 7 6 1
6 1 3
```
se afișează valoarea **2**.

Teza 3

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Valoarea variabilei **a** este un umăr natural. Care dintre expresiile C/C++ următoare are valoarea **1** dacă și numai dacă valoarea variabilei a are exact 3 cifre?
   a) `a/1000==0`
   b) `a/100!=0`
   c) `a/1000==0&&a/100!=0`
   d) `a/100==0&&a/10!=0`

2. Se consideră două tablouri unidimensionale A și B. Dacă A = (2, 7, 9, 11, 18), iar în urma interclasării lor în ordine descrescătoare se obține tabloul cu elementele (19, 18, 17, 12, 11, 9, 7, 7, 6, 2, 1), atunci B poate fi:
   a) (19, 18, 7, 6, 1)
   b) (19, 17, 12, 9, 7, 6, 2, 1)
   c) (19, 17, 12, 7, 6, 1)
   d) (19, 17, 12, 9, 7, 6, 1)

3. Variabilele a, b, n memorează numere naturale. Care dintre expresiile C/C++ următoare are valoarea **1** dacă și numai dacă rădăcina pătrată a lui n nu depășește valoarea variabilei a și nici valoarea variabilei b?
   a) `sqrt(n)<=a&&sqrt(n)>=b`
   b) `sqrt(n)<=a&&sqrt(n)<=b`
   c) `sqrt(n)<=a||sqrt(n)<=b`
   d) `sqr(n)<=a&&sqr(n)<=b`

4. Se consideră un tablou unidimensional în care elementele sunt, în această ordine 50, 49, 47, 23, 21, 17, 12, 10, 7, 5, 3. Pentru a afla indicele elementului din tablou cu valoarea x = 49, se aplică metoda căutării binare. Scrieți succesiunea corectă de elemente a căror valoare se compară cu valoarea lui x pe parcursul metodei indicate.
   a) 17, 47, 50, 49 **b)** 17, 47, 49 **c)** 17, 50, 49 **d)** 50, 49, 47

5. Se consideră algoritmul:
```
pentru i ←2, 10 execută
    dacă i%2=0 atunci
        scrie 1
    altfel
        scrie 2
```
   Ce se va afișa în urma execuției algoritmului?
   a) 121212121 **b)** 212121212 **c)** 2121212121 **d)** 121212121

SUBIECTUL II (40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

```
citește x
(x număr întreg)
y←0
pentru i←9, 0, –1 execută
    cx←x
    cât timp cx>0 execută
        dacă cx%10 = i atunci
            y← y*10 + i
        cx←cx/10
scrie y
```

a) Scrieți valoarea care se va afișa pentru x = 7172. **(6 puncte)**

b) Scrieți două seturi distincte de valori pentru x astfel încât rezultatul afișat să fie 4332. **(6 puncte)**

c) Scrieți programul C/C++ corespunzător algoritmului dat. **(10 puncte)**

d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască structura pentru cu alt tip de structură repetitivă. **(6 puncte)**

2. Se consideră declarările de mai jos, în care variabila `pre` memorează partea reală și variabila `pim` memorează partea imaginară a unui număr complex `z`. Scrieți o secvență de instrucțiuni C/C++ care citește de la tastatură informațiile despre numărul complex `z` și afișează pe ecran modulul numărului complex `z`. **(6 puncte)**

```
int pre, pim;
```

3. Variabilele **i** și **j** sunt de tip întreg. Scrieți expresia C/C++ care poate înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, să se afișeze numerele de mai jos, în această ordine. **(6 puncte)**

```
for(i=1;i<=5;i++)
  for(j=1;j<=5;j++)
    ............
```

```
2 1 1 1 1
1 2 1 1 1
1 1 2 1 1
1 1 1 2 1
1 1 1 1 2
```

SUBIECTUL III (30 de puncte)

1. Se citesc două numere naturale, `n` și `m`, se cere să se determine numărul de cifre utilizate pentru construirea numerelor naturale din intervalul `[n,m]`.
Exemplu: Dacă **n = 7** și **m = 56** atunci se va afișa valoarea **97.** Scrieți, în pseudocod, algoritmul de rezolvare al problemei enunțate. **(10 puncte)**

2. Scrieți un program C/C++ care citește de la tastatură numărul natural n (n ∈[2,100]), apoi cele n elemente ale unui tablou unidimensional, numere naturale din intervalul [0,10^9] știind

că elementele tabloului sunt ordonate crescător, să se elimine din vector un număr minim de elemente, astfel încât să nu existe în tablou două elemente alăturate identice. Să se afișeze pe ecran tabloul astfel obținut.
Exemplu: Dacă `n = 7` și elementele tabloului sunt `2 4 4 4 7 7 1` atunci pe ecran se va afișa `2 4 7 1`. **(10 puncte)**

3. Scrieți un program C/C++ care citește din fișierul text **BAC.TXT** un șir cu cel mult 100 000 de numere întregi formate din cel mult 2 cifre fiecare și afișează pe ecran separate printr-un spațiu, numărul sau numerele din fișier cu număr maxim de apariții.
Exemplu: Dacă în fișier sunt numerele `17 -8 13 17 -8 9 10 13` atunci pe ecran se vor afișa nu neapărat în această ordine `-8 13 17`.

a) Se cere să se proiecteze un algoritm eficient din punct de vedere al timpului de executare și al spațiului de memorie utilizat și să se realizeze o descriere de 3-4 rânduri a algoritmului ales justificându-se eficiența acestuia. **(2 puncte)**

b) Scrieți programul C/C++ corespunzător metodei descrise la punctul **a)**. **(8 puncte)**

Teza 4

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Variabilele **x, y** și **z** au valori numere naturale de cel mult 4 cifre. Care dintre expresiile C/C++ următoare are valoarea **1** dacă și numai dacă exact una dintre valori este impară?

a) `x%2 || y%2 || z%2`
b) `(x+y)%2==0 || (x+z)%2==0 || (y+z)%2==0`
c) `(x%10 + y%10 + z%10)%2>0`
d) `x%10%2 + y%10%2 + z%10%2==1`

2. Indicați cel mai mic și cel mai mare număr natural pe care îl poate memora variabila x, astfel încât expresia `sqrt(x+10)> x-10` să aibă valoarea 1.

a) 6 și 10 **b)** 0 și 14 **c)** 7 și 10 **d)** 0 și 10

3. Înlocuiți punctele de suspensie din secvența program C++ următoare, în ordinea scrierii lor în secvență, astfel încât să afișeze valoarea 1 dacă tabloul x format din n elemente este ordonat strict descrescător sau valoarea 0 în caz contrar.

```
int x[50],n,ordonat,i;
ordonat=...
for(i=2;i<=n;i++)
if(...)
   {ordonat=0;break;}
if(...)
  cout<<1;
else
  cout<<0;
```

a) 0, x[i]<=x[i+1], ordonat==1
b) 1, x[i]>x[i-1], ordonat==0
c) 1, x[i-1]<x[i], ordonat=1
d) 1, x[i-1]<=x[i], ordonat==1

4. Pentru tablourile unidimensionale **x** care are 3 elemente și **y** care are 4 elemente, se aplică operațiile următoare:
– interclasarea tablourilor x și y prin care se obține tabloul **z**;
– căutarea binară a valorii **t** în tabloul z. Valoarea t a fost găsită în tabloul **z** la a treia comparație, dacă s-au utilizat:

a) x = (5, 9, 24), y = (2, 14, 7, 10), t = 7
b) x = (2, 7, 14), y = (5, 9, 10, 24), t = 30
c) x = (2, 7, 14), y = (5, 9, 10, 24), t = 30
d) x = (2, 7, 14), y = (5, 9, 10, 24), t = 24

5. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

Completați punctele de suspensie astfel încât dacă se citește numărul 5, prin execuția acestui algoritm să se afișeze șase valori.

a) i = j **b)** i%2 = 0 **c)** i%2 = 1 **d)** i ≠ j

```
citește n (n număr natural nenul)
pentru i ←1, n execută
   pentru j←1, i execută
      dacă...
         scrie i + j
```

SUBIECTUL II (40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

a) Scrieți valoarea care se va afișa pentru n = 39. **(6 puncte)**
b) Scrieți toate valorile de o cifră ce pot fi date pentru n astfel încât rezultatul afișat să fie 1. **(6 puncte)**
c) Scrieți programul C/C++ corespunzător algoritmului dat. **(10 puncte)**
d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se utilizeze cel mult o structură repetitivă. **(6 puncte)**

```
citește n  (n număr natural nenul)
a←1
b←[n / 2]
cât timp a ≠ 0 și b>0 execută
   c←n
   cât timp c≥b execută
      c←c-b
   a←c
   b←b – 1
b←b+1
scrie b
```

2. Pentru memorarea rezultatelor la examenul de BAC obținute de 200 de candidați se consideră declarările C++ de mai jos:

```
int mat[200],lrom[200],inf[200];
```

Fiecare candidat are un cod de la 0 la 199, astfel candidatul cu codul i are la matematică nota **mat[i]**, la limba română nota **lrom[i]** și la informatică nota **inf[i]**. Scrieți o secvență de instrucțiuni C++ care determină și afișează numărul de candidați care au cel puțin nota 5 la fiecare probă și media generală cel puțin 7. **(6 puncte)**

3. În secvența de instrucțiuni de mai jos variabilele **i** și **j** sunt de tip întreg. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, să se afișeze caracterele de mai jos, în această ordine. **(6 puncte)**

```
for(i=0;i<=6;i++){
  for(j=0;j<=6;j++)
    ..............}
```

```
* - - - - - *
* * - - - * *
* - * - * - *
* - - * - - *
* - - - - - *
* - - - - - *
* - - - - - *
```

SUBIECTUL III (30 de puncte)

1. Un număr n se numește **rotund** dacă este par și în reprezentarea sa matematică în baza 2, nu sunt două cifre egale alăturate. De exemplu, numărul `12` este un număr rotund deoarece reprezentarea sa în baza 2 este `1010`. Numărul `18` nu este rotund deoarece reprezentat în baza 2 este `10010`. Scrieți un program în limbaj pseudocod care citește de la tastatură un număr natural nenul **n** cu cel mult 9 cifre și afișează valoarea 1 dacă n este număr rotund și 0 altfel. **(10 puncte)**

2. Scrieți un program C++ care citește de la tastatură un număr natural **n** ($2 \le n \le 10^3$), afișează pe ecran un număr **k** ce reprezintă numărul de valori prime cu n, mai mici decât n și un tablou unidimensional **p** ce conține cele k valori prime cu n, în ordine crescătoare. Spunem că un număr x este prim cu n dacă nu au divizori comuni mai mari decât 1.
Exemplu: Dacă `n=20` atunci `k=8`, iar `p=(1,3,7,9,11,13,17,19)`. **(10 puncte)**

3. Din fișierul text **BAC.TXT** se citesc, de pe prima linie un număr k ($n < 10^6$), iar de pe linia următoare cel mult 1 000 000 de numere naturale cu cel mult 2 cifre fiecare, separate prin câte un spațiu. Să se afișeze pe ecran numărul de pe a doua linie din fișier care va fi pe poziția k în șirul ordonat descrescător, dacă există sau –1 în caz contrar. Să se proiecteze un algoritm eficient din punct de vedere al timpului de executare.
Exemplu: Dacă fișierul conține numerele:
`6`
`21 9 16 2 16 2 9 4 9 21 9 2 8` atunci pe ecran se va afișa `9`.

a) Realizați o descriere de 3-4 rânduri a algoritmului ales justificând eficiența acestuia. **(2 puncte)**

b) Scrieți programul C/C++ corespunzător metodei descrise la punctul **a)**. **(8 puncte)**

Teza 5

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Variabilele a, b, c, d sunt întregi, $a \le b$, $c \le d$. Care dintre expresiile de mai jos are valoarea 1 dacă și numai dacă intersecția intervalelor (a, b) și [c, d] este vidă?
a) `c ≥ b && a < d`
b) `! (d ≥ a || c ≤ b)`
c) `! (b > c && a < d)`
d) `a ≥ d || b < c`

2. Indicați ce valoare are expresia C++ următoare: `abs(x)/pow(y,2)-pow(y,3)*pow(x+y,2)`, dacă toate variabilele din expresie sunt egale cu –1.
a) –3 **b)** 5 **c)** 2 **d)** –6

3. Variabila a memorează un număr natural care are exact 4 cifre. Care dintre expresiile C/C++ de mai jos are ca valoare numărul format din prima și ultima cifră a numărului memorat de a?
a) a/10 + a%100 **b)** a/1000 + a%10 **c)** a/1000*10 + a%10 **d)** a – a/10%10

4. Aplicând algoritmul de căutare binară pentru căutarea valorii x = 8 în tabloul unidimensional `(19,15,11,9,8,7,1)` se fac mai multe comparații. Care dintre următoarele valori este valoarea numărului de comparații?
a) 5 **b)** 2 **c)** 4 **d)** 3

5. Se consideră o secvență de program, în care s, p și a sunt variabile întregi. Completați punctele de suspensie astfel încât după execuția acestei secvențe variabilele s și p să fie egale.

```
s=0;p=1;a=2354;
while(a>0)
{if(...)
   s=s+a%10;
 else
   p=p*(a%10);
 a=a/10;
}
```

a) a%2==0 **b)** a%2%10==0 **c)** a%10==0 **d)** a%2!=0

SUBIECTUL II (40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y.

a) Scrieți valoarea care se va afișa dacă se citesc pe rând valorile 4, 8, 16, 10, 24. **(6 puncte)**

b) Pentru n = 3, dați exemple de trei numere naturale, nenule și distincte ce pot fi citite în variabila x, pentru care algoritmul să scrie 1. **(6 puncte)**

c) Scrieți programul C/C++ corespunzător algoritmului dat. **(10 puncte)**

d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să înlocuiți structura repetă până când cu o structură repetitivă condiționată anterior. **(6 puncte)**

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

2. Un număr de 100 de elevi au optat fiecare pentru unul dintre cercurile: informatică, pictură, robotică. Codul unui cerc este prima literă a denumirii cercului. Pentru evidența înscrierii la cercuri se utilizează un tablou unidimensional **cercuri** care memorează codul cercului pentru fiecare elev (**i** – informatică, **p** – pictură, **r** – robotică). Știind că cercuri[0] reprezintă opțiunea primului elev, scrieți o secvență de instrucțiuni C++ care determină numărul maxim de elevi care s-au înscris la un cerc. **(6 puncte)**

3. În secvența de instrucțiuni de mai jos, variabilele **i** și **j** sunt de tip întreg. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, să se afișeze numerele de mai jos, în această ordine. **(6 puncte)**

```
for(i=1;i<=6;i++){
  for(j=1;j<=6;j++)
    ..............}
```

```
1 1 1 1 1 1
1 2 2 2 2 1
1 2 3 3 2 1
1 2 3 3 2 1
1 2 2 2 2 1
1 1 1 1 1 1
```


SUBIECTUL III (30 de puncte)

1. Scrieți un program în limbaj pseudocod care citește de la tastatură un număr natural **n** de cel mult 6 cifre și o cifră nenulă c și afișează numărul maxim care se poate obține inserând în n cifra c.
Exemplu: Dacă **n = 57332 și c = 4** atunci, se va afișa numărul **574332.** **(10 puncte)**

2. Se citește de la tastatură un număr natural **n** ($2 \le n \le 1000$) și un tablou unidimensional `a`, care memorează un șir de `n` numere naturale, fiecare cu cel mult 2 cifre. Se elimină din tablou, pentru fiecare a_i, $1 \le i \le n$, elementele egale cu acesta, situate pe poziții consecutive, astfel încât, tabloul să conțină la final, un număr maxim de elemente distincte, în ordinea în care au apărut în șirul **a**. Scrieți un program C++ care afișează numărul de elemente rămase după eliminare și șirul modificat.
Exemplu: Dacă `n = 9` și tabloul **a** memorează valorile `1 2 2 2 3 3 5 8 9` atunci, se va afișa
5
`1 2 3 5 8`
(10 puncte)

3. Numim **interval asociat** unui șir de numere, perechea de numere naturale `a,b` (`a≤b`) cu proprietatea că șirul este format din toate valorile naturale distincte, cuprinse între `a` și `b` și scrise în ordine crescătoare. De exemplu, șirul `4 5 6 7 8` are intervalul asociat `4 8`, iar șirul `1 2 3 5 7 8 9` este format din trei subșiruri ale căror intervale asociate sunt `1 3`, `5 5` și `7 9`.
Scrieți un program C/C++ care citește din fișierul text **BAC.TXT** un șir S cu cel mult `1 000 000` de numere naturale din intervalul $[0, 10^3]$ și afișează pe ecran extremitățile intervalului care reprezintă intersecția tuturor intervalelor asociate, de lungime maximă, din care este format șirul dat sau mesajul multimea vida, în cazul în care acestea nu conțin niciun element comun.
Exemplu: Dacă fișierul conține numerele `2 3 4 1 2 3 4 5 6 2 3 4 5 0 1 2 3` atunci pe ecran vor fi afișate valorile `2 3` deoarece șirul este format din 4 intervale asociate `2 4`, `1 6`, `2 5` și `0 3` a căror intersecție este intervalul `2 3`. Dacă fișierul conține numerele `3 4 5 4 5 6 8` atunci pe ecran se va afișa `multimea vida`.

a) Se cere să se proiecteze un algoritm eficient din punct de vedere al timpului de executare și al spațiului de memorie utilizat și să se realizeze o descriere de 3-4 rânduri a algoritmului ales justificându-se eficiența acestuia. **(2 puncte)**

b) Scrieți programul C/C++ corespunzător metodei descrise la punctul **a)**. **(8 puncte)**

Teza 6

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Variabilele **x, y** și **z** sunt de tip întreg. Care dintre următoarele expresii C/C++ are valoarea 1 dacă și numai dacă cel mult două dintre variabilele precizate au valori egale?
   a) `!(x==y && x==z) || y!=z` **b)** `x==y || x==z || y==z`
   c) `(x-y)*(x-z)*(y-z)==0` **d)** `x==y==z`

2. Indicați ce condiție verifică variabila a de tip real din expresia C++ **sqrt(a)*10+pow(a,2)<100** astfel încât expresia să aibă valoarea 1.
   a) a > –5&&a <9 **b)** a > 3&&a <10 **c)** !(a < 0 ||a > = 9) **d)** !(a < 0&& a > = 9)

3. Fiind date 3 numere naturale nenule a, b, k (a ≤ b), numărul de numere din intervalul [a, b] divizibile cu k este:
   a) b/k – a/k **b)** b/k – a/k + a%k **c)** b/k – (a–1)%k **d)** b/k–a%k

4. Se dau vectorii a = (1, 3, 5), b = (2, 4, 6). Utilizând algoritmul de interclasare se obține vectorul c = (1, 2, 3, 4, 5, 6). Care este numărul de comparații care s-au făcut pe parcursul algoritmului?
   a) 5 **b)** 4 **c)** 6 **d)** 3

5. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

Ce se va afișa dacă se citesc numerele 5 3 2 22 9 21 81 63?

a) 0 **b)** 1 **c)** 2 **d)** 3

```
citește n, k, pk (n, k numere naturale nenule)
m←0
pentru i ←1, n execută
    citește a
    p←0
    cât timp a%k=0 atunci
        p←p+1
        a←[a/k]
    dacă p=pk atunci
        m←m+1
scrie m
```


SUBIECTUL II (40 de puncte)

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

a) Ce se va afișa dacă a = 8 și b = 12? **(6 puncte)**
b) Pentru a = 25, care este cea mai mare valoare pe care o poate primi b astfel încât să se afișeze doar '*'? **(6 puncte)**
c) Scrieți programul C/C++ corespunzător algoritmului dat. **(10 puncte)**
d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască „cât timp...execută" cu alt tip de structură repetitivă. **(6 puncte)**

```
citește a, b (a, b numere naturale, a≤b)
y←0
pentru x←b, a, –1 execută
    y←x
    k←0
    cât timp y>0 execută
        k←k+1–2*(y%2)
        y←[y/2]
    dacă k=0 atunci
        scrie x, ' '
scrie '*'
```

2. Se consideră tablourile unidimensionale **lungime**, **lățime** și **înălțime** formate din exact 100 de numere naturale fiecare ce memorează laturile a 100 de turnuri de formă paralelipiped. Știind că lungime[0], latime[0] și inaltime[0] reprezintă laturile primului turn scrieți o secvență de instrucțiuni C++ care determină și afișează pe ecran numărul de turnuri care au formă de cub. **(6 puncte)**

3. În secvența de instrucțiuni de mai jos, variabilele **i** și **j** sunt de tip întreg. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, să se afișeze numerele de mai jos, în această ordine. **(6 puncte)**

```
for(i=1;i<=6;i++){
  for(j=1;j<=6;j++)
    ............}
```

```
2 3 1 2 3 1
3 6 2 3 6 2
1 2 0 1 2 0
2 3 1 2 3 1
3 6 2 3 6 2
1 2 0 1 2 0
```

SUBIECTUL III (30 de puncte)

1. Scrieți un program în limbaj pseudocod care citește de la tastatură două numere naturale nenule a și b, care au cel mult 9 cifre, determină și afișează cel mai mic număr de permutări circulare la dreapta, cu câte o poziție, ale cifrelor lui a, astfel încât să se obțină numărul b. Dacă nu e posibil, se va afișa mesajul **Imposibil.**
   Exemplu: Pentru a = 120362 și b = 621203 se va afișa 2, iar pentru a = 732 și b = 237 se va afișa Imposibil. **(10 puncte)**

2. Se citește un număr natural **k** (**1≤k≤30**), un tablou unidimensional **v**, având **k** numere naturale în ordine strict crescătoare și un număr natural **n** (**1≤n≤99999**). Scrieți un program C/C++ care determină cel mai mic divizor prim al lui **n** și îl caută în vectorul **v**. Dacă divizorul prim va fi găsit, programul va afișa indicele elementului din vector egal cu el. În caz contrar, divizorul prim va fi adăugat în v pe poziția corespunzătoare astfel încât ordinea strict crescătoare să se păstreze și se va afișa vectorul v, obținut după inserare. În vectorul v, indicii elementelor se notează începând de la 1.
Exemplu: Dacă `k=4, v=(2,5,11,17), n=27` se va afișa **`v=(2,3,5,11,17)`**, iar dacă `k=4, v=(2,5,11,17), n=121`, atunci se va afișa **3.** **(10 puncte)**

3. Un număr este **palindrom** dacă citind cifrele numărului de la dreapta la stânga și de la stânga la dreapta se obține același număr. De exemplu 12321, 2002 sunt palindroame, iar 12312 nu este palindrom.
Fișierul text bac.txt conține pe prima linie n, un număr natural nenul, mai mic decât 10^5, iar pe a doua linie un șir de n numere din intervalul [1, 9], separate prin câte un spațiu. Se cere afișarea pe ecran a celui mai mare palindrom care se poate forma cu toate cifrele date. Dacă nu se poate construi un palindrom cu toate cifrele lui n, se va afișa –1. Proiectați un algoritm eficient din punct de vedere al memoriei utilizate și al timpului de executare.
Exemplu: Dacă fișierul **bac.txt** conține numerele:
`10`
`2 3 3 8 9 2 3 9 8 3` atunci pe ecran se afișează: `9833223389`.

a) Descrieți în limbaj natural algoritmul utilizat, justificând eficiența acestuia. **(2 puncte)**

b) Scrieți programul C/C++ corespunzător algoritmului descris la punctul **a)**. **(8 puncte)**

Teza 7

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Știind că variabila reală x aparține reuniunii intervalelor [3,7] și (20, 30) care dintre expresiile scrise mai jos, NU are valoarea 1?
a) !((x<3 ||x>7) && (x>=30 || x<=20))
b) (x<=7 && x>=3) && (x>20 && x<30)
c) (x<=7 && x>=3) || (x>20 && x<30)
d) !(x<3 || x>7) || !(x>=30 ||x<=20)

2. Indicați cel mai mic și cel mai mare număr natural pe care îl poate memora variabila a, astfel încât expresia `int(sqrt(a+100)) == 8+floor(a/10)` să aibă valoarea 1.
a) 40 și 49 **b)** 40 și 68 **c)** 44 și 49 **d)** 44 și 68

3. Ce valori au variabilele x și y, astfel încât prin interclasarea tablourilor (2, 5, 6, x, 30) și (1, y, 7, 20) să se obțină tabloul (1, 2, 2, 5, 6, 7, 10, 20, 30)?
a) x = 7, y = 2 **b)** x = 10, y = 7 **c)** x = 7, y = 10 **d)** x = 10, y = 2

4. Se consideră tabloul x = (128, 45, 602, 22, 7). Pentru fiecare element din vector se calculează suma cifrelor și se rearanjează elementele din tabloul x, astfel încât să fie ordonat crescător, în funcție de suma cifrelor lor. Care este tabloul obținut după ordonare?
a) (7, 22, 45, 128, 602)
b) (22, 7, 128, 45, 602)
c) (22, 7, 602, 45, 128)
d) (22, 45, 7, 128, 602)

5. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

Ce se va afișa dacă se citesc numerele 7 22 8 21 15 7 28 3?

a) 12 **b)** 14 **c)** 16 **d)** 18

```
citește n (n număr natural nenul)
s ←0; d←2
pentru i ←1, n execută
  citește a
  dacă a%d=0 atunci
    s←s+d
  altfel
    s←s+1
    d←d+1
scrie s
```

SUBIECTUL II (40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

```
citește n,y (y număr natural, y<10,
n număr natural nenul)
e←0
pentru i←1, n execută
    citește x (număr natural nenul)
    cx←x
    nr←0
    cât timp x≠0 execută
        dacă x%10=y
          atunci
           nr←nr+1
        x←[x/10]
    dacă nr%2=0 și nr>0
      atunci
      e←e+cx
scrie e
```

a) Scrieți ce se va afișa dacă se citesc, în această ordine, numerele 5, 3, 2334, 31, 1535, 330, 75. **(6 puncte)**

b) Scrieți un set de date de intrare care să determine afișarea valorii 1000. **(6 puncte)**

c) Scrieți programul C++ corespunzător algoritmului dat. **(10 puncte)**

d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască structura „pentru" cu alt tip de structură repetitivă. **(6 puncte)**

2. Se consideră tablourile latura1, latura2 și latura3 formate din cel mult 100 de numere naturale fiecare și un număr natural n ($2 \le n \le 100$). Știind că latura1[0], latura2[0] și latura3[0] reprezintă laturile primului triunghi și că latura 1[n – 1], latura 2[n – 1] și latura 3[n – 1] reprezintă laturile ultimului dintre cele n triunghiuri date, scrieți o secvență de instrucțiuni C++ care determină și afișează pe ecran valoarea maximă și valoarea minimă dintre perimetrele celor n triunghiuri date. **(6 puncte)**

3. În secvența de instrucțiuni de mai jos variabilele **i** și **j** sunt de tip întreg. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, să se afișeze caracterele de mai jos, în această ordine. **(6 puncte)**

```
for(i=0;i<6;i++){
  for(j=0;j<6;j++)
   ..............
    cout<<endl;}
```

```
A B C D E F
B A B B F B
C C A F C C
D D F A D D
E F E E A E
F F F F F A
```


SUBIECTUL III (30 de puncte)

1. Scrieți un program în limbaj pseudocod care citește de la tastatură un număr natural ($1 \le n \le 100$) și afișează termenul cu numărul de ordine egal cu **n** din șirul următor: 1, 2, **3,** 2, 1, 2, 3, 4, 5, **6,** 5, 4, 3, 2, 1, 2, 3, 4, 5, 6, 7, 8, **9,** 8, 7, 6, 5, 4, 3, 2, 1 etc.
Exemplu: Dacă **n = 10** atunci se va afișa numărul **6.** **(10 puncte)**

2. Scrieți un program C++ care citește de la tastatură un număr natural **n** ($1 \le n \le 100$) și un tablou unidimensional **x** format din **n** numere întregi și afișează produsul maxim obținut din două elemente distincte ale tabloului.
Exemplu: Pentru n = 6 și tabloul x = (–60, 36, –8, –2, 10, 5), se va afișa 480. **(10 puncte)**

3. Considerând un număr natural **x** ce conține cel puțin trei cifre, după ce se elimină prima și ultima cifră se obține un alt număr denumit **număr interior** al numărului x. Fișierul **bac.in** conține un șir având cel mult un milion de numere naturale care au cel puțin trei cifre și cel mult 5 cifre, separate prin câte un spațiu. Scrieți un program C++ care afișează pe ecran în ordine descrescătoare, separate prin spațiu, numerele interioare ale numerelor din fișier pentru care prima și ultima lor cifră sunt egale sau afișează mesajul Nu există, dacă fișierul nu conține niciun număr având prima și ultima cifră egale. Utilizați un algoritm eficient ca timp de execuție și spațiu de memorie utilizat.
Exemplu: Dacă fișierul conține șirul de numere 1151 234 2322 212 514 23122 atunci se vor afișa numerele 312 32 15 1.

a) Descrieți în limbaj natural algoritmul proiectat și justificați eficiența acestuia. **(2 puncte)**

b) Scrieți programul C/C++ corespunzător algoritmului descris. **(8 puncte)**

Teza 8

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Stabiliți care dintre următoarele expresii C++ are valoarea 1, dacă și numai dacă numerele naturale nenule x și y au aceeași paritate (ambele sunt numere pare sau ambele sunt numere impare).
a. x + y %2==0 **b)** (x + y) %2==0 **c)** (x*y) %2==0 **d)** (x*y) %2==1

2. Variabilele a, b, și c sunt de tip real. Indicați expresia C++ care reprezintă expresia aritmetică următoare: $\frac{(b+c)^2}{2a^2}$.

a) pow(b,2)+pow(c,2)/(2*pow(a,2))
b) pow(b,2)+pow(c,2)/2*pow(a,2)
c) pow(b+c,2)/(2*pow(a,2))
d) pow(b+c,2)/2*pow(a,2)

3. Se consideră un tablou x, format din 10 elemente, numere naturale și variabilele i și s, de tip întreg. Completați punctele de suspensie din secvența de program C++ următoare, astfel încât să se afișeze suma elementelor tabloului x, divizibile cu 2 și cu 7.

```
s=0;
for(i=0;i<10;i++)
 if(...)
  s=s+x[i];
cout<<s;
```

a) i%2==0 && i%7 ==0
b) x[i]%2==0 ||x[i]%7==0
c) x[i]%2!=0 && x[i]%7!=0
d) x[i]%2==0 && x[i]%7==0

4. Se consideră tabloul a = (10, –2, 4, 3, 1). Pentru a căuta valoarea x = 3 în vectorul a, în mod eficient se vor aplica algoritmii următori:
a) interclasare
b) cautare binară
c) ordonare și căutare binară
d) căutare secvențială

5. Se consideră o secvență de program, în care **i**, **j**, **n** și **s** sunt variabile întregi.

```
s=0;i=1;
do
{if(i+s<=n)
   {s=s+i;
    for(j=1;j<=i;j++)
    cout<<j<<" ";
    i++;
   }
else
s=n;
}
while(s<n);
```

Ce valoare are numărul **n** astfel încât la execuția acestei secvențe de program șirul numerelor afișate să conțină numărul 1 de patru ori?
a) 27 **b)** 16 **c)** 11 **d)** 4

SUBIECTUL II (40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

a) Scrieți valoarea care se va afișa pentru a = 3 și b =7. **(6 puncte)**
b) Scrieți un set de valori pentru a și b astfel încât rezultatul afișat să fie k = 8 și q = 4. **(6 puncte)**
c) Scrieți programul C++ corespunzător algoritmului dat. **(10 puncte)**
d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască prima structură „pentru“ cu alt tip de structură repetitivă. **(6 puncte)**

```
citește a, b (a ≤ b, a, b numere naturale)
k←0; q←0
pentru x←1, a execută
    pentru y←1, b execută
        dacă y%x=0
        atunci
            k←k+y
        altfel
            q←q+y
scrie k," ",q
```

2. Pentru evidența unor date despre 200 de orașe, se consideră două tablouri: **loc** ce memorează numărul de locuitori ai fiecărui oraș și **supr** ce memorează suprafața fiecărui oraș (exprimată în număr de km²). Orașele sunt notate cu numere naturale de la 1 la 200, denumite **coduri**. Astfel, orașul care are codul 1 are loc[1] locuitori și suprafața egală cu supr[1] km², orașul 2 are datele loc[2] și supr[2] etc.
Știind că densitatea populației reprezintă numărul de persoane care locuiesc pe un km², scrieți o secvență de instrucțiuni în limbajul C++ care să afișeze codurile tuturor orașelor care au o densitate mai mare de 15 locuitori/km². **(6 puncte)**

3. În secvența de instrucțiuni de mai jos variabilele **i** și **j** sunt de tip întreg. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, să se afișeze numerele de mai jos, în această ordine. **(6 puncte)**

```
for(i=0;i<5;i++){
  for(j=0;j<5;j++)
   { ..............
     cout<<endl;}
```

```
6  7  8  9  10
7  8  9  10 11
8  9  10 11 12
9  10 11 12 13
10 11 12 13 14
```

SUBIECTUL III (30 de puncte)

1. Scrieți un program în limbaj pseudocod care citește de la tastatură un număr natural **n** (**2≤n≤50**) și **n** numere naturale distincte, fiecare număr având cel mult **9** cifre și afișează numărul de perechi din șir, formate din numere introduse consecutiv, cu proprietatea că un număr este oglindit pentru celălalt număr.
   Exemplu: Dacă **n** = 7 și **v** = **(41, 14, 324, 423, 6, 82, 28)** atunci se va afișa valoarea **3**, deoarece șirul de numere conține 3 perechi de numere ce verifică proprietatea dată: (41, 14); (324, 423); (82, 28). **(10 puncte)**

2. Se citesc de la tastatură un număr natural **n** (**1≤n≤50**) și un tablou unidimensional **v** format format din n numere naturale nenule. Scrieți un program C++ care afișează cel mai mic factor prim comun tuturor elementelor din tabloul v notat **fp** și puterea factorului fp, notată cu **p**, pentru care numărul fp^p este divizor pentru fiecare element din tabloul v, cu p>0. Dacă elementele tabloului v nu au un factor prim comun atunci fp are valoarea –1 și p are valoarea 0.
   Exemplu: Pentru n = 4 și tabloul `v=(60,36,18,24)`, se va afișa `fp=2,p=1`, iar pentru n = 4 și tabloul `v=(22,60,21,25)`, se va afișa `fp=-1 și p=0`.
   (10 puncte)

3. Scrieți un program C++, eficient ca timp de execuție și spațiu de memorie utilizat, care afișează în fișierul **bac.txt**, în ordine strict crescătoare, pe câte o linie, toate palindroamele formate din 6 cifre impare. Primele 3 numere afișate în fișier sunt: 111111,113311,115511.
   a) Descrieți în limbaj natural algoritmul proiectat și justificați eficiența acestuia. **(2 puncte)**
   b) Scrieți programul C/C++ corespunzător algoritmului descris. **(8 puncte)**

Teza 9

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Stabiliți ce valori au variabilele întregi **n** și **x**, astfel încât la execuția secvenței de program următoare să se afișeze patru caractere * urmate de șase caractere #.

```
int i=1,n,x;
while(i*i<=n)
{cout<<'*'; i++;}
while(i<=n+x)
{cout<<'#';i++;}
```

   a) n = 18, x = 2 **b)** n = 5, x = 6 **c)** n = 16, x = –6 **d)** n = 24, x = 5

2. Completați punctele din suspensie din secvența de program următoare, astfel încât, prin execuția sa să determine prin variabila **divizor_maxim** cel mai mare divizor propriu al unui număr natural **n**, nenul.

```
int n,d,divizor_maxim=0;
for(d=...;d<=...;d++)
 if(n%d==0)
  if(...)
   divizor_maxim=d;
```

   a) `1,n/2,d>divizor_maxim`
   b) `2,n, d>divizor_maxim`
   c) `2,n/2, d>divizor_maxim`
   d) `2,sqrt(n), d<divizor_maxim`

3. Se consideră tabloul x = (23, 16, 10, 25, 17). Pentru fiecare element din vector se calculează numărul de divizori și se rearanjează elementele din tabloul x, astfel încât să fie dispuse în ordine descrescătoare în funcție de numărul lor de divizori. Care este tabloul obținut după ordonare?
   a) (25,23,17,16,10)
   b) (16,10,25,23,17)
   c) (17,23,25,10,16)
   d) (10,16,17,23,25)

4. Prin aplicarea algoritmului de interclasare a două tablouri a și b se obține tabloul t = (2, 2, 4, 5, 7, 10, 25, 40). Care sunt tablourile a și b pentru care s-a obținut tabloul t?
   a) `a=(2,4,7,10), b=(2,5,25,40)`
   b) `a=(2,4,5,10,25),b=(4,7,40)`
   c) `a=(4,7,25),b=(2,5,10,17,40)`
   d) `a=(2,4,25,10),b=(2,7,5,40)`

5. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și cu [x] partea întreagă a numărului real x.
Ce se va afișa dacă se citește numărul 35?

a) 35 **b)** 5 **c)** 7 **d)** 1

```
citește a (a număr natural nenul)
m ←0
pentru d ←a,2, –1 execută
    dacă  a%d=0
        dacă d%2=1
          atunci
            m ←d
          altfel
            m ←2*d
scrie m
```

SUBIECTUL II (40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

```
citește a,b,k (numere naturale
nenule, a≤b, k>1)
pentru x←a, b execută
  y ←0
  t←x
    cât timp t≠0 execută
    y←y*10+t%10
    t←[t/10]
    dacă y%k=0
      atunci
        scrie x," "
```

a) Scrieți ce se va afișa dacă se citesc numerele 12, 36, 3. **(6 puncte)**

b) Scrieți un set de date de intrare, cu proprietatea b – a = 2, pentru care se afișează trei numere. **6 puncte)**

c) Scrieți programul C++ corespunzător algoritmului dat. **(10 puncte)**

d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască structura „pentru" cu alt tip de structură repetitivă. **(6 puncte)**

2. Se consideră declarările de mai jos, prin care se memorează date despre 10 filme difuzate la un cinematograf pentru o singură zi.

```
int suma[11],suma_totala,suma_prec[11];
```

Pentru fiecare film se memorează: **suma** ce reprezintă suma totală încasată din vânzarea biletelor la data difuzării, și **suma_prec** ce reprezintă suma totală încasată din vânzarea

biletelor în ziua precedentă. **Codul** fiecărui film este egal cu numărul său de ordine din tabloul **suma**. Astfel suma[1] reprezintă suma încasată pentru filmul care are codul egal cu 1 și pentru care s-a încasat suma_prec[1] în ziua precedentă etc. Scrieți o secvență de instrucțiuni C++ care să determine și să afișeze pe ecran codurile filmelor care au avut încasări mai mari față de ziua precedentă și suma totală încasată de cinematograf, pentru toate filmele difuzate în ziua respectivă prin variabila `suma_totala`. **(6 puncte)**

3. În secvența de instrucțiuni de mai jos variabilele **i** și **j** sunt de tip întreg. Fără a utiliza alte variabile, scrieți una sau mai multe instrucțiuni care pot înlocui punctele de suspensie astfel încât, în urma executării secvenței obținute, să se afișeze numerele de mai jos, în această ordine. **(6 puncte)**

```
for(i=0;i<5;i++){
  for(j=0;j<5;j++)
   { ..............
     cout<<endl;}
```

```
0  1  2  3  4
2  3  4  5  6
0  1  2  3  4
6  7  8  9 10
0  1  2  3  4
```

SUBIECTUL III (30 de puncte)

1. Scrieți un program în limbaj pseudocod care citește un număr natural **n**, având cel mult 8 cifre și afișează două numere naturale **m** și **t**, unde **m** reprezintă cel mai apropiat termen din șirul lui Fibonacci mai mic decât **n** și **t** reprezintă cel mai apropiat termen din șirul lui Fibonacci, mai mare decât n.

Exemplu: Dacă n = 30 se afișează: m = 21, t = 34. **(10 puncte)**

2. Se consideră un număr natural **n** (**2≤n≤50**) și un tablou unidimensional **a**, format din **n** numere naturale nenule și distincte, fiecare număr având cel mult **9** cifre. Scrieți un program C++ care determină și afișează numărul de perechi distincte formate din elemente din tabloul a, prime între ele. Două numere naturale sunt prime între ele dacă au un singur divizor comun: numărul 1.

Exemplu: Dacă **n = 6** și **a = (42, 3, 14, 43, 8, 6)** atunci se va obține valoarea **7**, deoarece vectorul are 7 perechi de numere ce verifică proprietatea dată: (42, 43); (3, 14); (3, 43); (3, 8); (14, 43); (43, 8) ;(43, 6). **(10 puncte)**

3. Fișierul **bac.txt** conține pe prima linie un număr natural **n**, care este multiplu de 7 (7≤n≤70000) și pe a doua linie un șir de n numere întregi nenule, având cel mult trei cifre. Șirul de numere se împarte în secvențe de câte 7 numere denumite **benzi** și pentru fiecare bandă se determină cele mai mici două numere distincte **min1** și **min2**. Scrieți un program

C++, eficient ca timp de execuție și spațiu de memorie utilizat, care afișează pe ecran, pentru fiecare dintre benzi, în ordine crescătoare cele două valori **min1** și **min2** sau afișează numărul 0, dacă banda nu conține două valori distincte **min1** și **min2.**

Exemplu: Pentru **n = 21**, șirul de numere –3, 10, 1, 2, –5, 6, –5, 1, 1, 1, 1, 1, 1, 1, 10, 9, 1, 2, 3, 6, 7 se va afișa –5, –3, 0, 1, 2.

a) Descrieți în limbaj natural algoritmul proiectat și justificați eficiența acestuia. **(2 puncte)**

b) Scrieți programul C/C++ corespunzător algoritmului descris. **(8 puncte)**

Teza 10

SUBIECTUL I (20 de puncte)

Pentru fiecare dintre itemii de la 1 la 5, scrieți pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Variabilele **a, b** au valori numere naturale nenule, **a ≤ b.** Care dintre următoarele instrucțiuni C/C++ următoare atribuie variabilei intregi **c** numărul de valori pare din intervalul [a,b]?
   a) `c=(b-a+1)/2;`
   b) `c=a/2-b/2;`
   c) `c=b/2-(a-1)/2`
   d) `c=(a+b)/2;`

2. Care dintre următoarele expresii are valoarea 1 pentru orice număr natural x din intervalul [a,b], a≤b .
   a) `(x-a)*(x-b)>0`
   b) `(x-a)*(b-x)<0`
   c) `(x-a)*(x-b)<=0`
   d) `(a-x)*(b-x)>0`

3. Se consideră secvența de program C++ următoare.

```
int n,m,i,j,k;
for(i=1;i<=n;i++){
    for(j=1;j<=m;j++)
      if((i+j)%2==0)
        cout<<i+j;
    for(k=1;k<=i;k++)
        cout<<"*";
}
```

   Ce valoare au variabilele **n** și **m** astfel încât prin execuția programului ce conține această secvență să se afișeze `246*468**468***6810****`.
   a) n = 3, m = 5 **b)** n = 4, m = 4 **c)** n = 6, m = 4 **d)** n = 4, m = 6

4. Variabilele `x, y, a, b` memorează numere naturale. Care dintre următoarele expresii este adevărată dacă și numai dacă `[x,y]⊂[a,b]`.
   a) `a<=x||b<=y`
   b) `a<=x&&y<=b`
   c) `!(a<=x&&y<=b)`
   d) `x<=a&&b<=y`

5. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

Ce valoare se va afișa pentru x = 140 și y = 15?

a. 1 125 **b)** 8 1 **c)** 9 5 **d)** 10 4

```
citește x, y
(x, y numere naturale)
r←x
┌cât timp y ≤ r execută
│ r←r-y
└■
k←(x-r)/y
scrie k," ", r
```

SUBIECTUL II (40 de puncte)

1. Se consideră algoritmul alăturat, descris în pseudocod.

S-a notat cu x % y restul împărțirii numerelor întregi x și y și cu [x] partea întreagă a numărului real x.

a) Scrieți valoarea care se va afișa pentru n = 23 100. **(6 puncte)**

b) Scrieți cel mai mic și cel mai mare număr de două cifre care poate fi citit pentru n astfel încât algoritmul să afișeze valoarea 1. **(6 puncte)**

c) Scrieți programul C/C++ corespunzător algoritmului dat. **(10 puncte)**

d) Scrieți în pseudocod un algoritm echivalent cu cel dat, în care să se înlocuiască structura repetitivă „cat timp... execută" cu o structură repetitivă cu condiție finală. **(6 puncte)**

```
citește n (n număr natural, n>1)
p←1; m←0; x←2
┌cât timp n >= x execută
│┌dacă n%x=0 atunci
│ m←m+1; n←[n/x]
│ altfel
│  ┌dacă m%2 ≠ 0 atunci
│  │ p←p*x
│  └■
│ m←0; x←x+1
└■
┌dacă m%2 ≠ 0 atunci
│ p←p*x
└■
scrie p
```

2. Se consideră variabilele **a**, **b** și **c** ce memorează simultan lungimile laturilor unui triunghi (numere reale pozitive). Fără a utiliza variabile suplimentare, scrieți o expresie C++ care are valoarea 1 dacă și numai dacă trunghiul ale cărui laturi sunt memorate în variabilele **a**, **b** și **c** este un triunghi isoscel, fără a fi triunghi echilateral. **(6 puncte)**

3. Se consideră secvența următoare de program C++ în care se utilizează două tablouri unidimensionale **a** și **b**, formate din numere naturale. Tabloul a conține **n** numere pare în ordine crescătoare și tabloul b conține **m** numere impare în ordine crescătoare. Determinați două tablouri a și b, care să conțină un număr minim de elemente, astfel

încât secvența dată să afișeze 5 valori în ordine crescătoare, separate prin spațiu, alese alternativ din tablourile a și b. **(6 puncte)**

```
int i=0,j=0;
   while(i<n && j<m)
           if(a[i]<b[j])
             {cout<<a[i]<<" ";i++;}
           else
              {cout<<b[j]<<" "; j++;}
```

SUBIECTUL III (30 de puncte)

1. Se citește un număr natural n și un șir de n numere naturale cu cel mult trei cifre fiecare. Scrieți un algoritm în pseudocod care afișează, în ordinea citirii, numerele din șir cu proprietatea că reprezentarea lor în baza 2 conține doar cifre egale cu 1.
Exemplu: Pentru n = 5 și numerele 7 18 15 22 31 se vor afișa 7 15 31. **(10 puncte)**

2. Un tablou unidimensional se numește **k-palindrom** dacă după efectuarea a `k` permutări circulare cu o poziție spre stânga acesta devine palindrom (considerăm că un vector este palindrom dacă vectorul parcurs de la stânga la dreapta coincide cu vectorul parcurs de la dreapta la stânga). Se citește de la tastatură un număr natural `n` (`n≤100`), un tablou unidimensional `v` format din `n` numere întregi și un număr natural `k` (`1≤k<n`). Scrieți un program C++ care afișează mesajul DA dacă vectorul este k-palindrom sau mesajul NU în caz contrar.
Exemplu: Dacă n = 5, v = (2, 2, 4, 5, 4), k=1 se va afișa DA. **(10 puncte)**

3. Fișierul text **BAC.TXT** conține un șir S cu cel mult 100 000 de numere naturale din intervalul $[2, 10^9]$. Pentru fiecare valoare `x` din șir se determină numărul de cifre egale cu zero de la sfârșitul lui `x!` (**`unde x!=1*2*3...*x`**). Scrieți un program C/C++ care citește șirul S, obține și afișează pe ecran, cu spațiu între ele, numărul de valori **`x`** din șir care au un număr maxim de cifre egale cu zero la sfârșitul lui **`x!`** și care este această valoare maximă.
Exemplu: Dacă fișierul conține numerele **`102 12 50 100`** atunci pe ecran vor fi afișate valorile **`2 24`**.

a) Se cere să se proiecteze un algoritm eficient din punct de vedere al timpului de executare și al spațiului de memorie utilizat și să se realizeze o descriere de 3-4 rânduri a algoritmului ales justificându-se eficiența acestuia. **(2 puncte)**

b) Scrieți programul C/C++ corespunzător metodei descrise la **a)**. **(8 puncte)**

PARTEA a III-a

SOLUȚII

Indicații de rezolvare, răspunsuri, rezolvări complete

CAPITOLUL 1. Elemente de bază ale limbajului C++

1. !(a%2==0 || b%2==0) && !(a*b<0) ⇔ (a%2!=0 && b%2!=0) && (a*b>=0)
2. a>=n || b<=n && (n%3==0 && n%673==0)
3. x/10%10+x%10*10==x/100/10
4. -4
5. (5*4%7-19/(7-4%5))/2 este 0, deoarece (20%7-19/(7-4))/2)=(6-19/3)/2=(6-6)/3=0
6. 12, deoarece sunt 2 atribuiri n = 1023, k = 2, și alte 8 (câte două la fiecare iterație a instrucțiunii while)
7. Ordinea de execuție a celor 3 operații poate fi: 3, 1, 2 sau 2, 1, 3
   unde numerotăm operațiile astfel: **1)** x = y – x **2)** y = x + y **3)** y = y – x
8. Instrucțiunea C/C++ corespunzătoare următoarei operații de atribuire: $a \leftarrow \frac{(x+y^2)\cdot(y-1)}{x^2\cdot y}$
   este a=(x+y*y)*(y-1)/(x*x*y)
9. Expresiile C/C++ de mai jos au valoarea 1 dacă și numai dacă numărul memorat în variabila z este fie multiplu de x și de y, fie este divizor al lui x și al lui y.
   a) (z%x+z%y==0) || !(x%z || y%z)

CAPITOLUL 2. Algoritmi elementari

1. **a)** Răspuns: **32**. Se obține a=16=2^4, i=2, i*a=32
   b) Cea mai mică valoare de valoare de 3 cifre care se poate citi pentru **a = 103**, astfel încât valoarea afișată de algoritmul alăturat este egală cu 162 = 2*81 (3^4 = 81 și i = 2) și este maxim.

c) Algoritm echivalent:

```
citește a
i←0
a←a%10
┌dacă a>1 și a<10
│    atunci
│      ┌repetă
│      │   i←i+1
│      │   a←a*a
│      └până când a≥10
└■
scrie i*a
```

d) Program C/C++

```
#include <iostream>
using namespace std;
int main () {
    int a, i;
    cin>>a; i = 0;
    a = a % 10;
    while (a > 1 && a < 10) {
        i++;
        a=a*a;
    }
    cout << i * a;
    return 0; }
```

2. **a)** Răspuns: **32138.** Algoritmul construiește și afișează un număr de n cifre, format din prima cifră a fiecărui număr din șirul dat, în ordinea citirii.
   b) Pentru ca algoritmul să afișeze 9999, se va citi un șir de 4 numere, fiecare număr poate avea o cifră egală cu 9 sau poate avea 2 cifre, prima fiind egală cu 9.
   De exemplu: se pot citi 4 93 9 96 98 și se va afișa 9999.

c) Algoritm echivalent

```
citește n
a←0
┌pentru i←1, n execută
│      citește x
│  p ←1
│  ┌cât timp p<=[x/10] execută
│  │    p=p*10;
│  └■
│  a ←a*10+[x/p]
└■
scrie a
```

d) Program C/C++

```
#include <iostream>
using namespace std;
int main ()
{  int a,i;
  cin>>a;i=0;
  a=a%10;
  while(a>1&&a<10)
  {     i++;
        a=a*a;
  }
   cout<<i*a;
return 0; }
```

3. **a)** Răspuns: **13 23 17 19.** Algoritmul afișează perechile de numere prime între ele de forma (p, u), dintre numerele din intervalul [a, b], care sunt simetrice față de mijlocul intervalului.

b) **(4,9), (5,8), (6,7).**

c) Algoritm echivalent

```
citește a, b (numere naturale nenule, a<b)
p ← a;  u ← b
┌cât timp p<u  execută
│   x ← p;   y ← u
│   ┌cât timp y≠0 execută
│   │     z ← x%y
│   │     x ← y
│   │     y ← z
│   └■
│   ┌dacă x=1 atunci
│   │     scrie p, ' ', u, ';'
│   └■
│    p ← p+1;  u ← u–1
└■
```

d) Program C/C++

```
#include <iostream>
using namespace std;
int main() {
 int a, b, p, u, x, y, z;
 cin >> a >> b;
 p = a; u = b;
 while (p < u) {
   x = p; y = u;
   do {
     z = x % y;
     x = y;
     y = z;
   } while (y != 0);
   if (x==1)
     cout << p << " " << u <<" ";
   p = p + 1;  u = u – 1;
}
return 0;
}
```

4. **a)** Răspuns: **23.** Algoritmul realizează conversia numărului **a**, din baza **b** în baza 10 și afișează rezultatul conversiei efectuate.

b) a = 1057 și b = 8.

c) Algoritm echivalent

```
citește a, b (numere naturale nenule, 2≤b≤9)
p ← 1
x ← 0
┌cât timp a > 0 execută
│    x← x+(a % 10)*p
│    p ← p*b
│    a ← [a/10]
└■
scrie x
```

d) Program C/C++

```
#include <iostream>
using namespace std;
int main() {
  int a, b, m, x, p, i;
  cin >> a >> b;
  m = 0; x = 0;
  while (a > 0) {
   p = a % 10;
   for (i = 1; I <= m; i++)
       p = p * b;
   x = x + p;
   a = a / 10;
   m++;
 }
cout<<x;
return 0;}
```

5. **a)** Răspuns: 3

b) Valori pentru b: 1 5 25

c) Algoritm echivalent

```
citește n, a (număr natural nenul)
x ← 0
┌pentru i ←1, n execută
│   citește b (numere naturale nenule, a ≤ b)
│   p ← 1
│   ┌dacă p<b atunci
│   │    ┌repetă
│   │    │    p ← p*a
│   │    └■până când p ≥ b
│   └■
│   x ← x+[b/p]
└■
scrie x
```

d) Program C/C++

```
#include <iostream>
using namespace std;
int n, x, i, a, b, p;
int main() {
  cin >> n >> a;
  x = 0;
  for (i = 1; i <= n; i++) {
    cin >> b;
    p = 1;
    while (p < b)
      p = p * a;
    x = x + b / p;
  }
  cout << x;
  return 0;}
```

6. **a)** Răspuns: **9 4 2**

b) În intervalul [a, b] trebuie să existe doar un număr pătrat perfect de număr prim. De exemplu: a = 40 și b = 50.

c) Algoritm echivalent

```
citește x, y (numere naturale nenule, x≤y)
k ← 0
pentru y ←y, x, –1 execută
    d ← 2
    cât timp d*d≤y și y%d≠0 execută
        d ← d+1
    dacă d*d=y atunci
        scrie y, ' '
        k ← k+1
scrie k
```

d) Program C/C++:

```
#include <iostream>
using namespace std;
int x, y, k, d;
int main() {
  cin >> x >> y;
  k = 0;
  while (y >= x) {
    d = 2;
    while (d * d < y && y % d != 0)
          d++;
    if (d * d == y) {
      cout << y << , ,;
      k++;
    }
    y=y–1;
  }
  cout << k;
  return 0;}
```

7. **a)** Răspuns: **89 21 8 2**

b) Numărul n trebuie să fie cel mai mic număr de trei cifre ce este termen în șirul Fibonacci, **n=144**.

c) Algoritm echivalent

```
citește n (număr natural nenul, 2≤n)
k ← 0
cât timp n>0 execută
  a←1; b←1
  dacă b≤n atunci
    repetă
      b ← a+b
      p ← p*a
    până când b≥n
  scrie a, ' '
  n ← n–a
scrie k
```

d) Program C/C++:

```
#include <iostream>
using namespace std;
int n, k, a, b;
int main() {
  cin >> n;
  k = 0;
  while (n > 0) {
    a = 1; b = 0;
    while (b <= n). {
      b = a + b;
      a = b – a;
    }
    cout << a << , ,;
    n = n – a;
  }
  return 0;
}
```


8. **a)** Răspuns: **0**

b) Numărul n trebuie să fie palindrom de cel mult două cifre. Se va afișa 19.

c) Algoritm echivalent

```
citește n (număr natural )
p←1;  x←0;
dacă p < [n/p] atunci
  repetă
    x←x*10+[n/p] %10
    p←p*10
  până când p ≥ [n/p]
dacă x=[n/p] sau x=[n/(p/10)] sau n<10
  atunci  scrie 1
  altfel  scrie 0
```

d) Program C/C++

```
#include <iostream>
using namespace std;
int n, p, x, y, a, b;
int main() {
  cin >> n;  p = 1;  x = 0;
  while (p < n / p) {
    x = x * 10 + n / p % 10;
    p = p * 10;
  }
  if (x == n / p  ||  x == n / (p / 10)
     || n < 10)  cout << 1;
  else cout << 0;
  return 0; }
```

9. **a)** Răspuns: **42**

b) Numărul **n** trebuie să fie pătrat perfect de cel puțin trei cifre, de exemplu **100** sau **121**.

c) Algoritm echivalent

```
citește n (număr natural nenul, n>1 )
x ← 2; p←1
cât timp x≤n execută
  k←0;
  dacă n%x=0 atunci
    repetă
      n←[n/x]
      k←k+1
    până când b≥n
  dacă k%2≠0 atunci
    p←p*x
  x←x+1
scrie p
```

d) Program C/C++

```
#include <iostream>
using namespace std;
int n, p, k, x;
int main() {
  cin >> n;
  x = 2; p = 1;
  while (x <= n) {
    k = 0;
    while (n % x == 0) {
      n = n / x;
      k++;
    }
    if (k % 2 != 0)
      p = p * x;
    x++;
  }
  cout << p;
  return 0; }
```

CAPITOLUL 3. Fișiere text

1. Citim datele din fișier și aplicăm algoritmul de determinare a cmmdc și cmmmc.

```
#include <fstream>
using namespace std;
ifstream fin(„numere.in”);
ofstream fout(„numere.out”);
int n,i,a,b,ca,cb,r;
int main()
{   fin>>n;
    for(i=1; i<=n; i++)
    {
        fin>>a>>b;
        ca=a;cb=b;
        while(b)
        {
            r=a%b;a=b;b=r;
        }
        fout<<a<<' ,<<ca*cb/a<<endl;
    }
    return 0;
}
```

2. Citim succesiv valorile din fișier și le comparăm cu cele 3 variabile m1, m2, m3.

```
#include <fstream>
#include <iostream>
using namespace std;
ifstream fin(„numere.in”);
int x,m1,m2,m3;
int main()
{
    m1=m2=m3=1000000000;
    while(fin>>x)
    {
        if(x%3==0)
        {
            if(x<m1)
            m3=m2,m2=m1,m1=x;
            else if(x<m2)
            m3=m2,m2=x;
            else if(x<m3)m3=x;
        }
    }
cout<<m1<<' ,<<m2<<' ,<<m3;
return 0;
}
```

3. La fiecare pas prelucrăm două valori aflate pe poziții consecutive în fișier.

```
#include <fstream>
#include <iostream>
using namespace std;
ifstream fin(„numere.in”);
int x,y;
int main()
{
    fin>>x;cout<<x<<' ,;
    while(fin>>y)
    {
        if(y!=x)cout<<y<<' ,;
        x=y;
    }
    return 0;
}
```

4. Aplicăm algoritmul de determinare a lungimii maxime a unei secvențe în care elementele au o anumită proprietate, în acest caz divizibilitatea cu 10.

```
#include <fstream>
#include <iostream>
using namespace std;
ifstream fin("numere.in");
int x,lg,lgmax;
int main()
{
    while(fin>>x)
    {
        if(x%10==0)lg++;
            else
            {
                if(lg>lgmax)
   lgmax=lg;
                lg=0;
            }
    }
    if(lg>lgmax)lgmax=lg;
    cout<<lgmax;
    return 0;
}
```

5. Citim succesiv valorile din fișier, fiecare valoare o împărțim la 2 atât timp cât este divizibilă cu 2, dacă în final s-a obținut valoarea 1, afișăm valoarea inițială.

```
#include <fstream>
#include <iostream>
using namespace std;
ifstream fin("numere.in");
int x,cx;
int main()
{
    while(fin>>x)
    {
        cx=x;
        while(x%2==0)
            x=x/2;
        if(x==1)cout<<cx<<' ';
    }
    return 0;
}
```

6. Vom determina cea mai mică valoare cu toate cifrele egale mai mare decât a. Trecem succesiv prin toate valorile cu toate cifrele egale mai mici decât b.

```
#include <fstream>
#include <iostream>
#include <cmath>
using namespace std;
ofstream fout("bac.out");
int a,b,nra,na,i,p;
int main()
{
    cin>>a>>b;
    nra=log10(a)+1;
    na=0;
    for(i=1;i<=nra;i++)na=na*10+1;
    for(i=1;i<=9;i++)
        if(na*i>a)break;
    while(na*i<=b)
    {
        fout<<na*i<<' ';
        if(i==9)na=na*10+1,i=1;
        else i++;
    }
    return 0;
}
```

7. Parcurgem intervalul [a, b] și simulăm cerințele problemei.

```
#include <fstream>
#include <iostream>
using namespace std;
ofstream fout("bac.out");
int a,b,i,ci,si;
int main()
{   cin>>a>>b;
    for(i=a;i<=b;i++)
    {   ci=i;si=0;
        while(ci)
        {   si=si+ci%10;
            ci=ci/10;
        }
        if(i%si==i/si)
            fout<<i<<' ';
    }
    return 0;
}
```

8. Prefixele se obțin prin împărțiri succesive la 10.

```
#include <fstream>
#include <iostream>
using namespace std;
ofstream fout("bac.out");
int x;
int main()
{
    cin>>x;
    while(x)
    {
        fout<<x<<endl;
        x=x/10;
    }
    return 0;
}
```

9. Parcurgem numerele din fișier și determinăm numărul de ordine, până când găsim un număr divizibil cu 5 memorăm numărul și numărul de ordine al său.

```
#include <fstream>
#include <iostream>
using namespace std;
ifstream fin("numere.in");
int x,u,i,nr;
int main()
{

    while(fin>>x)
    {
        i++;
        if(x%5==0)
            nr=i,u=x;
    }
    cout<<u<<' '<<nr;
    return 0;
}
```

CAPITOLUL 4. Tablouri unidimensionale (vectori)

1.
```
#include <iostream>
#include<fstream>
using namespace std;
ifstream fin("numere.in");
int n,v[10000],i,aux,k;
int main()
{
    fin>>n;
    for(i=0;i<n;i++)
        fin>>v[i];
    k=n/2;
    aux=v[0];
    for(i=1;i<k;i++)v[i-1]=v[i];
    v[k-1]=aux;
    aux=v[n-1];
    for(i=n-2;i>=k;i--)
   v[i+1]=v[i];
    v[k]=aux;
    for(i=0;i<n;i++)
        cout<<v[i]<<' ';
    return 0;
}
```

2.
```
#include <iostream>
using namespace std;
int n,vf[10],i,a,b,m;
int main()
{
    cin>>n;
    do
    {
        vf[n%10]++;
        n=n/10;
    }
    while(n);
    for(int i=9; i>=0; i--)
        if(vf[i])
        {
            for(int j=1; j<=vf[i];
   j++)a=a*10+i;
             b=b*10+i;
        }
    cout<<a<<endl<<b;
    return 0;
}
```

3.
```
#include <fstream>
using namespace std;
ifstream fin("numere.in");
ofstream fout("numere.out");
int n,vf[100],i,x;
int main()
{
    while(fin>>x)vf[x]++;
    for(i=0; i<100; i++)
        while(vf[i])
        {
            fout<<i<<' ';
            vf[i]--;
        }
    return 0;
}
```

4.
```
#include <fstream>
using namespace std;
ifstream fin("numere.in");
ofstream fout("numere.out");
int n,vc[100],i,x;
int main()
{
    while(fin>>x)vc[x]=1;
    for(i=0; i<100; i++)
        if(vc[i])
        {
            fout<<i<<' ';
        }
    return 0;
}
```

5.
```
#include <fstream>
using namespace std;
ifstream fin("numere.in");
ofstream fout("numere.out");
int n,p[100],u[100],i,x,dmax;
int main()
{
    while(fin>>x)
    {
        i++;
        if(p[x]==0)p[x]=i;
        u[x]=i;
    }
    for(i=0; i<100; i++)
        if(p[i])
            fout<<i<<'
   '<<u[i]-p[i]<<endl;
    return 0;
}
```

6.
```
#include <fstream>
#include<iostream>
using namespace std;
ifstream fin("numere.in");
int n,v[1000],i,j;
int main()
{
    fin>>n;
    for(i=0;i<n;i++)
        fin>>v[i];
    for(i=0; i<n; i++)
        if(v[i]%2==0)
        {
            for(j=n-1; j>i; j--)
                v[j+1]=v[j];
            v[i+1]=2019;
            n++;i++;
        }
    for(i=0;i<n;i++)
            cout<<v[i]<<' ';
    return 0;
}
```

7.
```
#include <fstream>
#include<iostream>
using namespace std;
ifstream fin("numere.in");
int n,v[1000],i,j;
int main()
{
    fin>>n;
    for(i=0;i<n;i++)
        fin>>v[i];
    for(i=1; i<n; i++)
        if(v[i]==v[i-1])
        {
            for(j=i; j<n; j++)
                v[j]=v[j+1];
            n--;i--;
        }
    for(i=0;i<n;i++)
            cout<<v[i]<<' ';
    return 0;
}
```

8.
```
#include <fstream>
#include<iostream>
using namespace std;
ifstream fin("bac.txt");
int n,v[500],x,i,j,ok,ls,ld,nr,mij;
int main()
{
    fin>>n;
    for(i=0; i<n; i++)
        fin>>v[i];
    fin>>x;
    //aplicam algoritmul de cautare
   binara
    ls=0;
    ld=n-1;
    ok=0;
    while(ls<=ld)
    {
        mij=(ls+ld)/2;
        if(v[mij]<=x)
        {
            ok=1;
            nr=v[mij];
            ls=mij+1;
        }
        else ld=mij-1;
    }
    if(ok)cout<<nr;
    else cout<<"nu exista";
    return 0;
}
```

9.
```
#include <fstream>
#include<iostream>
using namespace std;
ifstream fin("bac.txt");
int n,a[500],b[500],i,j,m,k;
int main()
{
    cin>>m>>n;
    for(i=0; i<m; i++)
        cin>>a[i];
    for(j=0; j<n; j++)
        cin>>b[j];
    //aplicam algoritmul de
   interclasare
    i-m-1;
    j=n-1;
    while(i>=0&&j>=0)
    {
        if(a[i]>b[j])
        {
            if(a[i]%2!=0)
            {
                k++;
                cout<<a[i]<<' ';
            }
            i--;
        }
        else
        {
            if(b[j]%2!=0)
            {
                k++;
                cout<<b[j]<<' ';
            }
            j--;
        }
    }
    while(i>=0)
    {
        if(a[i]%2!=0)
        {
```

```
            k++;
            cout<<a[i]<<' ';
        }
        i--;
    }

    while(j>=0)
    {
        if(b[j]%2!=0)
        {
            k++;
            cout<<b[j]<<' ';
        }
        j--;
    }
    if(k==0)cout<<"nu exista";
    return 0;
}
```

CAPITOLUL 5. Tablouri bidimensionale (matrice)

1.
```
#include <iostream>
using namespace std;
int n, a[21][21],i,j,x;
int main()
{
    cin>>n>>x;
    for(i=1; i<=n; i++)
        for(j=1; j<=n; j++)
            if(i==1||j==1)a[i][j]=x;
            else a[i][j]=a[i-1]
   [j]+a[i][j-1];
    for(i=1; i<=n; i++)
    {
        for(j=1; j<=n; j++)
            cout<<a[i][j]<<' ,;
        cout<<endl;
    }
    return 0;
}
```

2.
```
#include <iostream>
using namespace std;
int m,n,a[11][11],i,j,maxl,s;
int main()
{
    cin>>m>>n;
    for(i=1; i<=m; i++)
        for(j=1; j<=n; j++)
            cin>>a[i][j];
    for(i=1; i<=m; i++)
    {
        maxl=a[i][1];
        for(j=2; j<=n; j++)
            if(a[i][j]>maxl)
   maxl=a[i][j];
        s=s+maxl;
    }
    cout<<s;
    return 0;
}
```

3.
```
#include <iostream>
using namespace std;
int m,n,a[11][11],i,j,minc,s;
int main()
{
    cin>>m>>n;
    for(i=1; i<=m; i++)
        for(j=1; j<=n; j++)
            cin>>a[i][j];
    for(j=1; j<=n; j++)
    {
        minc=a[1][j];
        for(i=2; i<=m; i++)
            if(a[i][j]<minc)
   minc=a[i][j];
        s=s+minc;
    }
    cout<<s;
    return 0;
}
```

4.
```
#include <iostream>
using namespace std;
int n,a[10][10],i,j;
int main()
{
    cin>>n;
    for(j=1; j<=n; j++)
            cin>>a[1][j];
    for(i=2; i<=n; i++)
    {
        for(j=1; j<n; j++)
            a[i][j]=a[i-1][j+1];
        a[i][n]=a[i-1][1];
    }
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
            cout<<a[i][j]<<' ,;
        cout<<endl;
    }
    return 0;
}
```


5.
```
#include <iostream>
using namespace std;
int n,a[10][10],x,i,j,nr;
int main()
{
    cin>>n;
    for(i=1; i<=n; i++)
        for(j=1; j<=n; j++)
            cin>>a[i][j];
    cin>>x;
    for(i=2; i<n; i++)
    {
        if(a[i][1]%x==0)
            nr++;
        if(a[i][n]%x==0)
            nr++;
        if(a[1][i]%x==0)
            nr++;
        if(a[n][i]%x==0)
            nr++;
    }
    if(a[1][1]%x==0)nr++;
    if(a[1][n]%x==0)nr++;
    if(a[n][1]%x==0)nr++;
    if(a[n][n]%x==0)nr++;
    cout<<nr;
    return 0;
}
```

6.
```
#include <iostream>
using namespace std;
int m,n,a[51][51],i,j,ok,afis;
int main()
{
    cin>>m>>n;
    for(i=1; i<=m; i++)
        for(j=1; j<=n; j++)
            cin>>a[i][j];
    for(i=1; i<=m; i++)
    {
        ok=1;
        for(j=2; j<=n&&ok; j++)
            if(a[i][j]!=a[i][1])
   ok=0;
        if(ok)
        {
            cout<<i<<' ,<<a[i]
   [1]<<endl;
            afis=1;
        }
    }
    if(!afis)cout<<"nu exista";
    return 0;
}
```

7.
```
#include <iostream>
using namespace std;
int m,n,a[51][51],i,j,ok,nr;
int main()
{
    cin>>m>>n;
    for(i=1; i<=m; i++)
        for(j=1; j<=n; j++)
            cin>>a[i][j];
    for(j=2; j<=n; j++)
    {
        ok=1;
        for(i=1; i<=m&&ok; i++)
            if(a[i][j]==a[i][1])
   ok=0;
        if(ok)nr++;
    }
    cout<<nr;
    return 0;
}
```

8.
```
#include <iostream>
using namespace std;
int n,a[16][16],i,j,x=1;
int main()
{
    cin>>n;
    for(i=1; i<=n; i++)
        for(j=1; j<=n; j++)
            if(i%2!=0)a[i][j]=x++;
            else a[i][n+1-j]=x++;
    for(i=1; i<=n; i++)
    {
        for(j=1; j<=n; j++)
            cout<<a[i][j]<<' ,;
        cout<<endl;
    }
    return 0;
}
```

9.
```
#include <iostream>
using namespace std;
int n,a[16][16],i,j,x=1;
int main()
{
    cin>>n;
    for(i=1; i<=(n+1)/2; i++)
    {
        for(j=i; j<=n+1-i; j++)
            a[i][j]=x++;
        for(j=i+1; j<=n+1-i; j++)
            a[j][n+1-i]=x++;
        for(j=n-i; j>=i; j--)
```

```
            a[n+1-i][j]=x++;
        for(j=n-i; j>i; j--)
            a[j][i]=x++;
    }
    for(i=1; i<=n; i++)
    {
        for(j=1; j<=n; j++)
            cout<<a[i][j]<<' ,;
        cout<<endl;
    }
    return 0;
}
```

CAPITOLUL 6. Șiruri de caractere

1. Varianta 1

```
#include <fstream>
#include<cstring>
using namespace std;
ifstream fin(„sir.in");
ofstream fout(„sir.out");

int main()
{char s[201],c[51],sn[201]="",*p;
int nr_cuv=0,i,j;
fin.getline(s,201);fin.
getline(c,51);
for(i=0;i<strlen(s);i++)
    if(s[i]==c[0]&& (i==0||s[i-1]==' ,))
{
    nr_cuv++;
    j=i+1;
    while(s[j]!=' ,&&j<strlen(s))
        j++;
    strcpy(s+i,s+j); //eliminarea
cuvantului cu prefix comun din text
}
fout<<s<<"\n"<<nr_cuv<<"\n";
    return 0;
}
```

Varianta 2

```
#include <fstream>
#include<cstring>
using namespace std;
ifstream fin(„sir.in");
ofstream fout(„sir.out");

int main()
{char s[201],c[51],sn[201]="",*p;
int nr_cuv=0;
fin.getline(s,201);fin.
getline(c,51);
p=strtok(s," „);
while(p)
{
    if(p[0]==c[0])//prima litera
reprezinta un prefix al sirului
        nr_cuv++;
    else
        strcat(sn,p);
    strcat(sn," „);
    p=strtok(NULL," „);
}
sn[strlen(sn)-1]=NULL;
strcpy(s,sn);
fout<<s<<"\n"<<nr_cuv<<"\n";
    return 0;
}
```

2.

```
#include <iostream>
#include<cstring>
using namespace std;
int main()
{ char t[51],tnou[160]="",unu[]=
"unu",doi[]="doi",aux[51];int i;
cin>>t;
for(i=0;i<strlen(t);i++)
 {
     if(t[i]=='1')
{strcpy(aux,t+i+1);
strcpy(t+i+1,unu);
strcat(t,aux);
}
     if(t[i]=='2')
{strcpy(aux,t+i+1);
strcpy(t+i+1,doi);
strcat(t,aux);
}
 }
cout<<t;
    return 0;
}
```

3.

```
#include <iostream>
#include<cstring>
#include<cctype>
using namespace std;
int n=0;

void numar(char s[])// adauga cifrele
din sirul s la numarul n
{int i;
    for(i=0;i<strlen(s);i++)
    if(isdigit(s[i])!=0)
        n=n*10+(s[i]-'0');
}
int main()
{char s1[251],s2[251];
 cin>>s1>>s2;
 numar(s1);numar(s2);
 cout << n;
 return 0;
}
```

4.

```
#include <fstream>
#include<cstring>
using namespace std;
ifstream fin(„cuvinte.in");
ofstream fout(„cuvinte.out");

int main()
{char prim[31],linie[31],ultim[31];
int n,i;

fin>>n; fin>>linie;
strcpy(prim,linie);strcpy(ultim,linie);
for(i=1;i<n;i++)
{
    fin>>linie;
    if(strcmp(linie,prim)<0)
        strcpy(prim,linie);
    if(strcmp(linie,ultim)>0)
        strcpy(ultim,linie);

}
fout<<prim<<"\n"<<ultim;
return 0;

}
```

5.

```
 #include <iostream>
#include<cstring>
using namespace std;
int main()
{ char t[201],tnou[201],*cuv,
separatori[]=",.?!",voc[]="aeiou";
 cin.getline(t,201);strcpy(tnou,"");
 cuv=strtok(t,separatori);
 while(cuv)
 {
  if(strchr(voc,cuv[0])!=NULL&&strchr
(voc,cuv[strlen(cuv)-1])!=NULL)
    {strcat(tnou,cuv);strcat(tnou," „);}
     cuv=strtok(NULL,separatori);
 }
    if(strlen(tnou)>0)
        {tnou[strlen(tnou)-
1]=NULL;cout << tnou;}
    else
        cout<<"Nu exista";
    return 0;
}
```

6.

```
#include <iostream>
#include<cstring>
#include<cctype>
using namespace std;

char e[201],operatori[]="-*/",
paranteze[]="[(„;
int main()
{int i,j,corect=1;
cin>>e;
if(strlen(e)<3)
    corect=0;
if(!islower(e[0])&&!isdi-
git(e[0])&&!strchr(operatori,e[0])&&
!strchr(paranteze,e[0]))
       corect=0;
for(i=1;i<strlen(e)&&corect==1;i++)
  {if(!islower(e[i])&&!isdi-
git(e[i])&&!strchr(operatori,e[i])&&
!strchr(paranteze,e[i]))
       corect=0;

if(islower(e[i])&&islower(e[i-1]))
    corect=0;

if(isdigit(e[i])&&isdigit(e[i-1]))
    corect=0;
   if(islower(e[i])&&isdi-
git(e[i-1])||islower(e[i-1])&&
isdigit(e[i]))
      corect=0;
    if(strchr(operatori,e[i])&&
strchr(paranteze,e[i-1]))
       corect=0;
   if(strchr(operatori,e[i])&&
strchr(operatori,e[i-1]))
    corect=0;
  }
    cout <<corect;
    return 0;
}
```

7.

```
#include<iostream>
#include<cstring>

using namespace std;

int main()
{char voc,c;int i,n,lg=0,lgmax=0;
cin>>voc;cin>>n;
i=0;
while(i<n)
{cin>>c;i++;
    if(c==voc)
      {lg=1;break;}
 }
while(i<n)
{
    cin>>c;
     lg++;
     if(c==voc)
  {
      lgmax+=lg;
      lg=0;
  }
i++;
}

if(lgmax>=2)
    cout<<lgmax;
else
    cout<<"Nu exista";
    return 0;
}
```

8.

```
#include <iostream>
#include<cstring>
using namespace std;

int main()
{char s[401],*p,cuv[201],vocale[]=
"aeiouAEIOU";int nr_ap=0,i;
cin.get(s,401);
p=strtok(s," ");
while(p!=NULL &&nr_ap==0)
{
    for(i=0;i<strlen(p)-1;i++)
        if(strchr(vocale,p[i])!=0
&&strchr(vocale,p[i+1])!=0)
    {
        nr_ap++;strcpy(cuv,p); break;
    }
    p=strtok(NULL," ");
}
while(p!=NULL)
{
    if(strcmp(cuv,p)==0)
        nr_ap++;
    p=strtok(NULL," ");
}
if(nr_ap==0)
       cout << "Nu exista" << endl;
    else
        cout<<cuv<<" "<<nr_ap;
    return 0;
}
```

9.

```
#include <fstream>
#include<cstring>
using namespace std;
ifstream fin("cuvinte.in");
ofstream fout("cuvinte.out");

int main()
{char cuv[41][21],linie[21]; int
n,i,j,nr_cuv=0,lg;
//cuvintele citite de lungime lg se
memoreaza in tabloul cuv
fin>>n>>lg;
for(i=1;i<=n;i++)
{
    fin>>linie;
    if(strlen(linie)==lg)
    {
        nr_cuv++;
        strcpy(cuv[nr_cuv],linie);
    }
}
//ordonare alfabetica a cuvintelor
for(i=1;i<nr_cuv;i++)
    for(j=i+1;j<=nr_cuv;j++)
     if(strcmp(cuv[i],cuv[j])>0)
     {strcpy(linie,cuv[i]);
      strcpy(cuv[i],cuv[j]);
      strcpy(cuv[j],linie);
      }
 for(i=1;i<=nr_cuv;i++)
        fout<<cuv[i]<<"\n";
return 0;
}
```


CAPITOLUL 7. Structuri de date neomogene

1.

```
struct melodie
{ char nume[30],autor[20], tara[20];
   int loc_top,an_top;
}m[200];
```

a)

```
int i;
for(i=0;i<200;i++)
if(strcmp(m[i].tara, "Romania")==0 &&
   m[i].loc_top<4 && m[i].
   an_top==2018)
   cout<<m[i].nume<<endl;
```

b) // sortarea vectorului m, după câmpul nume

```
int i,j; melodie aux;
for(i=0; i<199;i++)
 for(j=i+1; i<200;i++)
   if(strcmp(m[i].nume,m[j].nume)>0)
     {strcpy(aux, m[i]);
      strcpy(m[i], m[j]);
      strcpy(m[j],aux);
     }
for(i=0;i<200;i++)
   cout<<m[i].nume<<endl;
```

c) //sortarea vectorului m, după câmpul țara

```
  int i,j; melodie aux;
  for(i=0; i<199;i++)
     for(j=i+1; i<200;i++)
      if(strcmp(m[i].tara,m[j].
   tara)>0)
      {strcpy(aux, m[i]);
       strcpy(m[i], m[j]);
       strcpy(m[j],aux);
      }
 else
if(strcmp(m[i].tara,m[j].tara)==0 &&
   m[i].loc_top>m[j].loc_top)
     {strcpy(aux, m[i]);
      strcpy(m[i], m[j]);
      strcpy(m[j],aux);
     }
 for(i=0;i<200;i++)
   cout<<m[i].nume<<" "<<m[i].
   tara<<endl;
```

2.

```
  struct medalie
{char nume[20],judet[30],
   nume_medalie[10];
 int an;
 }sp[500];
int i,nr_aur=0,nr_ag=0,nr_bronz=0;
for(i=0;i<500;i++)
     if(sp[i].an==2019 &&
strcmp(sp[i].judet,"Iasi")==0)
       {if(strcmp(sp[i].
nume_medalie,"aur")==0)
      nr_aur++;
        else
      if(strcmp(sp[i].
nume_medalie,"argint")==0)
           nr_ag++;
       else
           nr_bronz++;
       }
```

3.

```
struct data
  {int zi,luna an;
  };
struct consultatie
  { char pacient[20],cabinet[40];
  data data_cons;
  }c[100];
int i,nr=0;
for(i=0;i<100;i++)
  if(c[i].data_cons.zi==10&&
   c[i].data_cons.luna==8&&
   c[i].data_cons.an==2019&&
   strcmp(c[i].cabinet,
   "stomatologie")==0)
      {nr++;
       cout<<c[i].pacient<<endl;
       }
cout<<nr;
```

4.

```
struct data
     {int zi,luna an;
   };
 struct excursie
   {char nume[30], oras[40],tara[30] ;
    data data_exc;
   }e[300];
char numed[30];
int i,nr=0;
for(i=0;i<300;i++)
  if(strcmp(numed, e[i].nume)==0)
   {nr++;
    cout<<e[i].tara<<" "<<e[i].
   oras<<" "<<e[i].data_exc.zi<<" ";
    cout<<"<<e[i].data_exc.luna<<"
   "<<e[i].data_exc.an<<endl;
}
if(nr==0)
 cout<<"Nicio excursie";
```

5.
```
struct angajat
{char nume[30], functie[20],
   departament[30];
 int salariu_baza, ora_sp, nr_ore_sp,
   total_sporuri,salariu_obt;
 }a[300];
int i,sal_max=0,nr_total=0;
```
a)
```
   for(i=0;i<300;i++)
a[i].salariu_obt=a[i].salariu_baza+
   a[i].total_sporuri+
a[i].ora_sp*a[i].nr_ore_sp;
```
b)
```
  for(i=0;i<300;i++)
  if(strcmp(a[i].
departament,"productie")==0)
   if(a[i].salariu_obt>sal_max)
    sal_max= a[i].salariu_obt;
cout<<sal_max;
```
c.
```
   for(i=0;i<300;i++)
  if(strcmp(a[i].
departament,"marketing")==0)
    nr_total+= a[i].nr_ore_sp;
cout<<nr_total;
```

6.
```
struct student
{ char nume[20],grupa[10],bursa[3];
  int an_studiu, nr_credite;
}st[300];

int i;
```
a)
```
for(i=0;i<300;i++)
  if(st[i].nr_credite>=30)
   strcpy(st[i].bursa,"Da");
 else
   strcpy(st[i].bursa,"Nu");
```
b)
```
for(i=0;i<300;i++)
 if(strcmp(st[i].bursa,"Da")==0)
  cout<<st[i].nume<<" "<<st[i].an_
studiu<<" "<<st[i].grupa;
```

7.
```
struct examinare
{char
   tip_examinare[10],disciplina[20];
  int nr_credite;
};

struct student
{ char nume[20],grupa[10];examinare
   e[10];
  int an_studiu, nr_total_credite;
}st[200];
int i,j,max_credite=0,nr_max;
```
a)
```
for(i=0;i<200;i++) //i -indice
   student
  for(j=0;j<10;j++)//j -indice
   disciplina

    st[i].nr_total_credite+=
   st[i].e[j].nr_credite
```
b)
```
for(i=0;i<200;i++)
   for(j=0;j<10;j++)
      if(st[i].e[j].nr_credite>0 &&
       strcmp(st[i].e[j].tip_
       examinare,"proiect")==0)
       { cout<<st[i].nume<<"
"<< st[i].an_studiu;
        cout<<" "<<st[i].grupa;
cout<<st[i].e[j].disciplina;
        }
```
c)
```
  for(i=0;i<200;i++)
    if(st[i].
  nr_total_credite>max_credite)
   {max_credite= st[i].
  nr_total_credite;
    nr_max=1;
    }
     else
     if (st[i].nr_total_credite==
        max_credite)
      nr_max++;
     cout<<nr_max;
```

8.
```
struct data
     {int zi,luna an;
   };
struct spectacol
 { char denumire[30],autor[20];data
   data_sp;
   int nr_bilete, pret;
 }s[50];
int suma=0,i;
```
a)
```
   for(i=0;i<50;i++)
   if(s[i].data_sp.luna==5&& s[i].
   data_sp.an==2019)
    suma+=s[i].nr_bilete*s[i].pret;
  cout<<suma;
```
b)
```
     for(i=0;i<50;i++)
   if(strcmp(s[i].
   autor,"I.L.Caragiale")==0)
     {cout<<s[i].data_sp.zi<<"
   "<< s[i].data_sp.luna<<" ";
     cout<< s[i].data_sp.zi;
   }
```


9.
```
struct data
   {int zi,luna an;
  };
  struct conferinta
 { char denumire[30],oras[20],
  tara[25],tematica[20];
   data data_conf; char nume[20],
  tip_inreg[20];
 } c[100];
int i;
```
a)
```
 for(i=0;i<100;i++)
if(strcmp(c[i].tip_inreg,"lector")==0
&&c[i].data_conf.an==2019)
if(strcmp(c[i].tematica,"IT")==0)
if(strcmp(c[i].tara,"Romania")==0 ||
strcmp(c[i].tara,"SUA")==0 ||
strcmp(c[i].tara,"Japonia")==0)
  cout<<c[i].nume<<endl;
```
b)
```
 int nr=0,i;
for(i=0;i<100;i++)
  if(c[i].data_conf.an==2019&& c[i].
data_conf.luna==4)
if(strcmp(c[i].tematica,
"medicina")==0)&&strcmp(c[i].
oras,"Iasi")==0&&strcmp(c[i].
tip_inreg,"participant")==0)
{nr++;
  cout<<c[i].nume<<endl;
}
cout<<nr;
```

CAPITOLUL 8. Subprograme

1.
```
int suma_cifre(int n)
{int s=0,f,i;
while(n)
{f=1;
 for(i=2;i<=n%10;i++)
   f=f*i;
 s=s+f;
 n=n/10;
}
return s;
}
```

2.
```
int baza(int n,int b)
{
 while(n)
{if(n%10>=b)
  return 0;
 n=n/10;
}
return 1;
}
```

3.
```
int palindrom(int n)
{
    int inv=0,cn=n;
    while(cn)
    {
        inv=inv*10+cn%10;
        cn/=10;
    }
    if(inv==n)
        return 1;
    else
        return 0;
}
```

4.
```
void dpalindrom(int a,int b)
{
    int cx,x,nr,p;
    for(x=a;x<=b;x++)
    {
        cx=x/100;nr=0;p=1;
        while(cx>=100)
            {nr=nr+(cx%10)*p;
            cx/=10;p=p*10;
            }
        if(palindrom(nr)==1)
            cout<<x<<" „;
    }

}
```

5.
```
void suma(int n,int v[],int &suma)
{ int i,cx,s;suma=0;
  for(i=1;i<=n;i++)
   { cx=v[i]; s=0;
     while(cx>9)
     { s=s+cx%10;
       cx=cx/10;
     }
if (cx==s)
suma+=v[i];
}
if(suma==0)
suma=-1;}
```

6.
```
void cmmdc(int a,int b,int &cmd)
{int r;
do
 {r=a%b;
   a=b; b=r;}
}
while(r!=0);
cmd=a;
}
```

```
void prime(int n, int y,int x[],int p[],
int &np)
{int i,c;np=0;
for(i=1;i<=n;i++)
{cmmdc(x[i],y,c);
if(c==1)
   {np++;
    p[np]=x[i];
   }
}
```

```
7. void egale(int n,int x[],int y[],
int &eg)
{int i=1,j=1;eg=0;
 while(i<=n && j<=n)
 {if(x[i]<y[j])
   i++;
  else
   if(x[i]==y[j])
    {eg++;i++;j++;}
  else
    j++;}}
```

```
8. void interschimbare(int n,int m,
int linie1,int linie2,int a[40][40])
{int j,aux;
  for(j=0;j<m;j++)
    { aux=a[linie1][j];
     a[linie1][j]= a[linie2][j];
     a[linie2][j]=aux;
    }
}
```

```
9. struct punct
    {float x,y;};

float distanta(punct A,punct B)
{float D;
 D=sqrt(pow(A.x-B.x,2)+pow(A.y-B.y,2));
 return D;
}
```

CAPITOLUL 9. Funcții recursive

1. Funcția determină cel mai mare divizor comun prin împărțiri repetate a 2 numere transmise ca parametru. Valoarea returnată de funcție este 40.
2. Funcția determină cel mai mare divizor comun prin scăderi repetate a 2 numere transmise ca parametru. Valoarea returnată de funcție este 40.
3. Funcția determină a^b. Valoarea returnată de funcție este 1024.
4. Funcția determină a^b prin algoritmul de exponențiere rapidă (ridicare la putere în timp logaritmic). Valoarea returnată de funcție este 1024.
5. Funcția afișează scrierea ca puteri ale lui 2 a unui număr natural n transmis ca parametru. Se va afișa 1 4 8 32.
6. Funcția transformă un număr din baza 10 în baza 16. Funcția va afișa C6.
7. Funcția va afișa 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1
8. Funcția va calcula C_n^k. Valoarea returnată de funcție la apelul f(5, 3) este 10.
9. Funcția va afișa 1 2 3 3 2 3 3.

CAPITOLUL 10. Backtracking. Elemente de combinatorică

1. Valoarea afișată este 14. (1 + 2 + 3 + 6, 1 + 2 + 4 + 5, 1 + 2 + 9, 1 + 3 + 8, 1 + 4 + 7, 1 + 5 + 6, 1 + 11, 2 + 3 + 7, 2 + 4 + 6, 2 + 10, 3 + 4 + 5, 3 + 9, 4 + 8, 5 + 7)
2. Numărul de steaguri este $A_6^3 = 120$, ultimul steag construit are culorile verde, roșu și portocaliu.
3. Numărul de posibilități de alegere este $C_5^3 = 10$. Algoritmul folosit este cel de generare al combinărilor. Ultima echipă este Robert, Teodora și Vlad.
4. Pentru n = 4 se construiesc în ordine lexicografică submulțimile {}, {1}, {1, 2}, {1, 2, 3}, {1, 2, 3, 4}, {1,2,4}, {1,3}, {1,3,4}, {1,4}, {2}, {2,3}, {2,3,4}, {2, 4}, {3}, {3, 4}, {4}. A cincea submulțime este {1, 2, 3, 4}.
5. Se vor genera în ordine (a, b, c), (a, b, d), (a, b, e), (b, b, c), (b, b , d), (b, b, e). Al patrulea element generat teste (b, b, c).
6. Se vor genera doar primele 2 cifre, celelalte 2 se obțin prin oglindire. Avem 9*9=81 numere palindrom. Dintre aceste numere sunt palindrom doar cele care încep cu o cifră pară, astfel sunt 4*5=20 numere palindrom pare.
7. Sunt 6 permutări care încep cu 5 4, a 7-a permutare este 5 3 4 2 1.
8. Putem a asocia fiecărei partiții un vector care indică numărul submulțimii din care face parte elementul astfel generăm vectorii: 1 1 1 1, 1 1 1 2, 1 1 2 1, 1 1 2 2, 1 1 2 3, 1 2 1 1, 1 2 1 2, 1 2 1 3, 1 2 2 1, 1 2 2 2, 1 2 2 3, 1 2 3 1, 1 2 3 2, 1 2 3 3, 1 2 3 4. Numărul de partiții formate din exact 2 submulțimi este 7.
9. Submulțimile care conțin secvența 2 4 nu pot conține elementul 3, conțin obligatoriu 2 4 și putem adăuga doar 2 elemente din mulțimea {1, 5, 6}. Numărul de soluții este $C_3^2 = 3$ (1 2 4 5, 1 2 4 6, 2 4 5 6).

CAPITOLUL 11. Elemente de teoria grafurilor

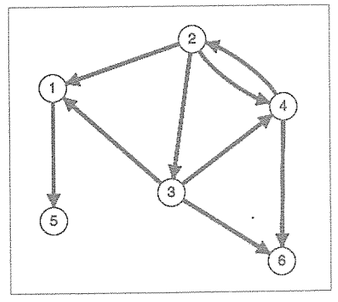
Grafuri neorientate

1. a) Gradul maxim este 5 și nodul 4 este singurul nod de grad maxim.
   b) Listele de adiacență ale grafului dat.
   L1: 2, 4; L2: 1, 3, 4, 5; L3: 2, 5, 8;
   L4: 1, 2, 6, 7, 8; L5: 2, 3, 8, 9; L6: 4, 7;
   L7: 4, 6; L8: 3, 4, 5, 9; L9: 5, 8;
   c) Un ciclu elementar de lungime maximă: 1, 2, 3, 5, 9, 8, 4, 1.
   d) Graful este conex, dar are două noduri de grad impar, neadiacente. Numărul minim de muchii de eliminat astfel încât să devină eulerian este 2: [4,8] și [8,3].
   e) Pentru ca un graf complet să fie eulerian trebuie ca numărul de noduri din graf să fie impar, astfel încât toate gradele să fie pare. Astfel, numărul de subgrafuri subgrafuri complete, fără vârfuri izolate este: $C_9^3 + C_9^5 + C_9^7 + C_9^9$.
   f) Linia corespunzătoare nodului 4 din matricea de adiacență: 1 1 0 0 0 1 1 1 0.
   g) Numărul de muchii ce trebuie adăugate grafului astfel încât să devină graf complet = 9*8/2 - 14 = 22.
   h) Componentele conexe ale subgraful indus de vârfurile de grad impar sunt: C1 = {3, 5, 9}; C2 = {7}; C3 = {1}.
2. G un graf neorientat cu 30 de vârfuri reprezentat printr-o matrice de adiacență cu 24 de valori nenule $\Rightarrow$ m = 24/2 = 12; $\Rightarrow$ Graful poate fi format din cel puțin 17 și cel mult 25 componente conexe.
3. G un graf cu 25 de muchii, fără noduri izolate. Numărul maxim de noduri din graf este 50.
4. Numărul de grafuri neorientate cu 10 noduri care se pot forma astfel încât nodul 1 să fie izolat = $2^{C_9^2}$.
5. Numărul minim de noduri = 11 (2 sub grafuri complete cu 2 vârfuri și o muchie + o componentă conexă cu 16 muchii și 7 vârfuri).
   Numărul maxim de noduri = 36 (2 subgrafuri complete cu 2 vârfuri și un subarbore cu 14 muchii și 15 noduri).
6. Numărul de subgrafuri euleriene cu 3 vârfuri din G este C_8^3.
7. Numărul total de muchii din G = 20.
8. Nodurile se vor dispune într-un lanț de lungime n – 1, deci puterea maximă va fi 1 + 2 + 3 + ... (n - 1) = n(n - 1)/2

Graful este bipartit și conține 2 cicluri de lungime pară A = {1, 5, 6, 7} și B = {2, 3, 4, 8}. Numărul de muchii de adăugat pentru a fi complet este 4*4 - 8 = 8.

Grafuri orientate

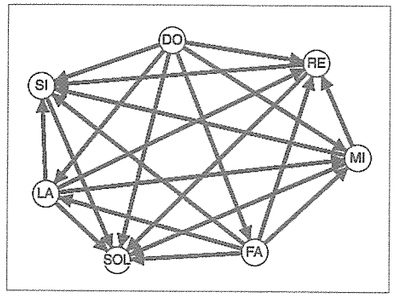
1. a) $d^+(2) = 1, d^-(2) = 2$,
   b) $L_5 = \{2, 3, 7\}$
   c) $D_{14} = \{1, 3, 5, 7, 6, 4\}$
   d) 1 0 0 0 1 1 0 0
   e) $d^+(3) = 1, d^-(3) = 3$, $d^+(4) = 1, d^-(4) = 2$, $d^+(7) = 1, d^-(7) = 2$, $d^+(6) = 1, d^-(6) = 2$,
   f) $C_1 = \{1, 3, 5, 6, 7\}$; $C_2 = \{4\}$; $C_1 = \{8\}$
   g) Numărul minim de arce de adăugat pentru ca graful alăturat să devină tare conex este 1. De exemplu: putem adăuga arcul (6,8) și graful devine tare conex.
   h) $C_1 = \{1, 2, 1\}$; $C_2 = \{3, 5, 3\}$; $C_3 = \{1, 3, 5, 2, 3\}$; $C_4 = \{4, 7, 6, 4\}$; $C_5 = \{3, 5, 7, 6, 3\}$; $C_5 = \{3, 5, 7, 4, 6, 3\}$
2. Numărul maxim de vârfuri izolate se obține determinând numărul minim de noduri al unui subgraf care conține toate arcele. Numărul de arce din graf este 50 pentru că suma gradelor interioare este egală cu numărul de arce. Determinăm cel mai mic k, astfel încât $k(k - 1) \geq 22$. Numărul maxim de vârfuri izolate este 22 = 30 - 8.



3. Determinăm cel mai mic k, astfel încât $k(k-1) \geq 20$ $\Rightarrow k = 5$. Numărul minim de vârfuri din graf este 5, iar numărul maxim este 40 și se obține punând fiecare arc între două noduri diferite.
4. $3^{C_{10}^3} = 3^{45} = 9^{15}$.
5. Cele 18 arce se distribuie în 6 componente tare conexe, formate fiecare din 3 noduri și 3 arce, dispuse în circuit.
6. Graful conține drumuri elementare de forma x_1, x_3x_5, ..., x_k unde $1 \leq k \leq 99$, impar și x_2, x_4x_6, ..., x_k, unde $1 \leq k \leq 99$, par. Numărul de componente

tare conexe este 100, iar numărul minim de arce de adăugat pentru a fi tare conex este 2: arcele (99,2) și (100,1).

7. L_{DO}: FA, LA, MI, RE,SI, SOL;
   L_{FA}: LA, MI, RE, SI, SOL;
   L_{LA}: MI, RE, SI, SOL;
   L_{MI}: RE, SI, SOL;
   L_{RE}: SI, SOL;
   L_{SI}: SOL;
   L_{SOL}: -;

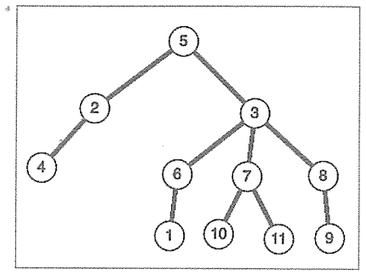


   Drum elementar prin toate nodurile: DO, FA, LA, MI, RE, SI, SOL.

8. Graful orientat complet **G** cu **10** vârfuri are cel puțin **11 arce** (un circuit elementar prin toate vârfurile) și cel mult **90** arce (graf plin).

9. $\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

Arbori

1. Pentru arborele din figura 14:
   a) Frunzele din arbore sunt: 3, 5, 7, 8 și 9.
   b) Arborele are înălțimea 3.
   c) Nodurile ce pot fi alese ca rădăcină astfel încât orice vârf să aibă cel mult doi fii sunt: 3, 5, 6, 7, 8, 9.
   d) Ascendenții nodului 7 sunt: 1, 2 și 6.
   e) Lungimea maximă a unui ciclu elementar care se poate forma în arborele dat prin adăugarea unor muchii este egală cu 9, ciclul va conține cele 9 noduri din arborele inițial. Un exemplu de astfel de ciclu este: 1, 2, 5, 7, 6, 3, 8, 9, 4, 1.
2. Graful este aciclic, deci fiecare componentă va conține un număr de muchii egal cu numărul de noduri −1, deci graful dat are 3 + 4 + 6 = 13 muchii.
3. Numărul minim de noduri din arbore este egal cu 22.
4. 

   a) Arborele este reprezentat sus.
   b) Înălțimea maximă a arborelui este 5 și se obține dacă se alege ca rădăcină unul dintre nodurile: 1, 4, 9, 10 sau 11. Înălțimea minimă a arborelui este 3 și se obține dacă se alege ca rădăcină nodul 5.
   c) Rădăcina = 3 ⇒ tata = (6, 5, 0, 2, 3, 3, 3, 3, 8, 7, 7).
   d) Lungimea maximă a unui lanț elementar din acest arbore este egală cu 5 și există 4 lanțuri elementare de lungime maximă.
5. Un graf complet cu 6 noduri are 15 muchii. Graful va deveni arbore dacă se vor elimina 10 muchii.
6. Un lanț elementar din arbore care unește nodurile 16 și 20 este 16, 8, 4, 2, 5, 10, 20.
7. Numărul de noduri din arbore este cel mult 1023 și cel puțin 9.
8. Arborele are 16 noduri și înălțimea maximă a arborelui este egală cu 5.
9. Descendenții nodului 8 din arbore sunt: 2, 3, 7 și 9.

Indicații de rezolvare, răspunsuri, rezolvări complete

TEZE

SPECIALIZAREA MATEMATICĂ-INFORMATICĂ

TEZA 1

Subiectul I

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

Subiectul II

1. **a)** algoritmul calculează a^n se va afișa 128

b) putem alege orice valoare pentru a și n = 0 (a = 3, n = 0), sau a = 1 și orice valoare pentru n (a = 1, n = 7)

c)
```
#include<iostream>
using namespace std;
int a,n,p;
int main()
{ cin>>a>>n;
p=1;
while(n>0)
{ if(n%2==0)
        {a=a*a;n=n/2;}
  else
        {p=p*a;n=n-1;}
}
cout<<p;
return 0;
}
```

d) Vom utiliza structura repetitivă „repetă – până când”

```
citește a, n
(a număr întreg, n număr natural)
p←1
dacă n>0 atunci
┌repetă
│  dacă n%2=0 atunci
│     a←a*a
│     n←[n/2]
│  altfel
│     p←p*a
│     n←n-1
└■
└până când n=0
scrie p
```

2.
```
if(e.dn.an==2000)cout<<e.nume;
else cout<<e.dn.zi<<' '<<e.
dn.luna<<' '<<e.dn.an;
```

3.
```
A[i][j]=(i+j)%5
```

Subiectul III
```
#include <iostream>
#include<cstring>
using namespace std;
char s[101],cs[101],*p;
int lmax;
int main()
{
     cin.get(s,101);
     strcpy(cs,s);
     //determinam lungimea maxim[ a
     unui cuvant
     p=strtok(s,"#");
     while(p)
     {if(strlen(p)>lmax)lmax=strlen(p);
     p=strtok(NULL,"#");
     }
     //inversam cuvintele de lungime
     maxima
     p=strtok(cs,"#");
     while(p)
     {if(strlen(p)==lmax)strrev(p);
     cout<<'#'<<p;
     p=strtok(NULL,"#");
     }
     cout<<'#';
     return 0;
}
```

1. Determinăm cel mai mic element din b și apoi numărăm câte elemente din a sunt strict mai mici decât acesta.
```
int numarare(int n, int m, int a[],
int b[])
{
     int nr=0,i,bmin=b[0];
     for(i=1;i<m;i++)
          if(b[i]<bmin)bmin=b[i];
     for(i=0;i<n;i++)
          if(a[i]<bmin)nr++;
     return nr;
}
```

2. **a)** Vom citi numerele din fișier succesiv și vom compara la fiecare pas prima cifră a numărului curent cu ultima cifră a numărului anterior, dacă cele două cifre sunt egale vom crește lungimea secvenței curente, altfel comparăm lungimea secvenței curente cu lungimea maximă, actualizând valoarea acesteia dacă este cazul. Algoritmul este

eficient din punct de vedere al timpului de execuție deoarece este un algoritm liniar și din punct de vedere al spațiului de memorie deoarece utilizează doar variabile simple.

b)
```
#include <iostream>
#include<fstream>
using namespace std;
ifstream fin(„bac.txt”);
int x,y,lg,lgmax,pc;
int main()
{
   fin>>x;lg=1;
   while(fin>>y)
   {
      pc=y;
      while(pc>9)
      pc=pc/10;
      if(pc==x%10)lg++;
      else
      {
        if(lg>lgmax)
          lgmax=lg;
        lg=1;
      }
       x=y;
       }
   cout<<lgmax;
   return 0;
   }
```

TEZA 2

Subiectul I

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

Subiectul II

1. **a)** Algoritmul numără câte dintre valorile citite sunt puteri ale lui 2. Pentru valorile date se va afișa 3, deoarece sunt 3 puteri ale lui 2 (4, 8, 16)

b) Oricare patru numere care nu sunt puteri ale lui 2. Exemplu: 12 15, 13, 45

c)
```
#include <iostream>
using namespace std;
int n,x,y;
int main()
{
    cin>>n;
    while(n>0)
    {
        cin>>x;
        while(x%2==0)
            x=x/2;
        if(x==1)
            y++;
        n--;
    }
    cout<<y;
    return 0;
}
```

d) Vom utiliza structura repetitivă „pentru – execută”

citește n (n număr natural)

y←0

pentru i←1, n execută
  citește x
  cât timp x%2=0 execută
    x←x/2
  dacă x=1 atunci
    y←y+1

scrie y

2.
```
cin>>p.v[i].x>>p.v[i].y;
```

3.
```
if(i+j<6)A[i][j]=1;
else if(i+j==6)A[i][j]=3;
else A[i][j]=2;
```

Subiectul III

1. Vom utiliza doi vectori de frecvență, unul care memorează frecvența literelor mari și unul care memorează frecvența literelor mici, determinăm maximul din cei doi vectori de frecvență și afișăm literele cu frecvența egală cu frecvența maximă.

```
#include <iostream>
#include<cstring>
using namespace std;
int fa[26],fA[26],fmax,i;
char s[101];
int main()
{
    cin.get(s,101);
    for(i=0; i<strlen(s); i++)
        if(s[i]!=' ')
        {
            if(s[i]>='a'&&s[i]<='z')
                fa[s[i]-'a']++;
            else fA[s[i]-'A']++;
        }
    for(i=0; i<26; i++)
    {
        if(fa[i]>fmax)fmax=fa[i];
        if(fA[i]>fmax)fmax=fA[i];
    }
    for(i=0; i<26; i++)
    {
        if(fa[i]==fmax)cout<<(char)
        (i+'a')<<' ';
        if(fA[i]==fmax)cout<<(char)
        (i+'A')<<' ';
    }
    cout<<fmax;
    return 0;
}
```

2. Numerele naturale care au exact trei divizori sunt pătratele perfecte de numere prime. Vom număra câte pătrate perfecte de numere prime sunt în intervalul [a, b].

```
int numarare(int a,int b)
{
    int nr=0,i,j,prim;
    a=ceil(sqrt(a));b=floor(sqrt(b));
    // odata modificate a si b este
    suficient
    //sa numaram numerele prime din
    noul interval
    for(i=a;i<=b;i++)
      {
          prim=1;
          if(i<=1)prim=0;
          for(j=2;j*j<=i&&prim;j++)
                if(i%j==0)prim=0;
          if(prim)nr++;
      }
    return nr;
}
```

3. Deoarece valorile din cele două șiruri sunt numere naturale cu cel mult două cifre, vom utiliza doi vectori de frecvență, un vector pentru șirul a și un vector pentru șirul b. Apoi vom determina min{fa[i]/fb[i], cu proprietatea că fb[i]!=0, i=0,99}. Algoritmul este eficient din punct de vedere al timpului de execuție deoarece este un algoritm liniar.

b)
```
#include <iostream>
#include<fstream>
using namespace std;
ifstream fin("bac.txt");
int fa[100],fb[100],i,x,nrs,n,m;
int main()
{
    fin>>n>>m;
    for(i=1;i<=n;i++)
    {fin>>x;
    fa[x]++;
    }
    for(i=1;i<=m;i++)
    {fin>>x;
    fb[x]++;
    }
    nrs=0;
    for(i=0;i<=99;i++)
        if(fb[i]!=0)
            if(nrs==0||fa[i]/fb[i]<nrs)
            nrs=fa[i]/fb[i];
    cout<<nrs;
   return 0;
}
```

TEZA 3

Subiectul I

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

Subiectul II

1. Algoritmul determină numărul de cifre egale cu 0 din reprezentarea numărului în baza 2.

a) se va afișa 1 ($13_{10}=1101_2$)

b) 7, 15 (orice număr de forma 2^n-1)

c)
```
#include <iostream>
using namespace std;
int nr,x;
int main()
{
    cin>>x;
    while(x>0)
    {
        if(x%2==0)
            nr++;
        x=x/2;
    }
    cout<<nr;
    return 0;
}
```

d) Vom utiliza structura „cât timp – execută”

citește x (x număr natural)

nr←0

dacă x>0 atunci
  repetă
    dacă x%2=0 atunci
    nr←nr+1
    x←[x/2]
  până când x=0

scrie nr

2.
```
if(E1.dn.luna<E2.dn.luna)cout<<E1.nume;
else if(E1.dn.luna==E2.dn.luna)
if(E1.dn.zi<E2.dn.zi)cout<<E1.nume;
else cout<<E2.nume;
else cout<<E2.nume;
```

3. bacalaureat a i

Subiectul III

1.
```
#include <iostream>
using namespace std;
int a[52][51],n,i,j,k;
int main()
{
    cin>>n;
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
            cin>>a[i][j];
    k=n/2+1;
    for(i=k;i<n;i++)
        for(j=1;j<=n;j++)
        a[i][j]=a[i+1][j];
    for(j=k;j<n;j++)
        for(i=1;i<=n;i++)
        a[i][j]=a[i][j+1];
    n--;
```

```
    for(i=1;i<=n;i++)
    {    for(j=1;j<=n;j++)
            cout<<a[i][j]<<' ,;
        cout<<endl;
    }
    return 0;
}
```

4. Construim un vector local care memorează cifrele impare ale lui n. Ordonăm descrescător vectorul construit și construim numărul cu cifrele din vector.

```
void numar(int n,int &m)
{
    int v[11],k,i,j;
    k=0;
    while(n)
    {
        if(n%2==1)
        {
            k++;
            v[k]=n%10;
        }
        n=n/10;
    }
    for(i=1; i<k; i++)
        for(j=i+1; j<=k; j++)
            if(v[i]<v[j])
   swap(v[i],v[j]);
    m=0;
    for(i=1; i<=k; i++)
        m=m*10+v[i];
if(m==0)m=-1;
}
```

4. **a)** Vom determina cel mai mic multiplu comun al numerelor n și m. Afișăm toți multiplii acestuia care au exact k cifre printr-un procedeu liniar, mergand din multiplu în multiplu. Algoritmul este eficient din punct de vedere al spațiului de memorie utilizând doar variabile simple. Algoritmul este eficient din punct de vedere al timpului de execuție deoarece nu parcurge secvențial toate numerele de k cifre, ci doar numerele care trebuie afișate.

b)

```
#include <iostream>
#include<fstream>
using namespace std;
ofstream fout(„bac.txt");
int n,m,cn,cm,multiplu, mic,
mare,r,i,k;
int main()
{
    cin>>n>>m>>k;
    cn=n;cm=m;
    while(m)
    {r=n%m;n=m;m=r;}
    multiplu=cn*cm/n;
    mic=1;
    for(i=1;i<k;i++)mic=mic*10;
    mare=mic*10-1;
    //determinam in mic cel mai mic
    numar de k
    //cifre divizibil cu multiplu
    if(mic%multiplu!=0)mic=(mic/
    multiplu+1)*multiplu;
    while(mic<=mare)
    {
        fout<<mic<<' ,;
        mic=mic+multiplu;
    }
    return 0;
}
```

TEZA 4

Subiectul I

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

Subiectul II.

1. Algoritmul construiește cel mai mare număr care se poate forma utilizând toate cifrele lui x.

a) Se va afișa 7721.

b) Două numere care conțin cifrele 2, 3, 4 de același număr de ori pot fi 3324, 2343.

c)

```
#include<iostream>
using namespace std;
int x,y,cx,i;
int main()
{cin>>x;
 y=0;
 for(i=9;i>=0;i--)
{
      cx=x;
      while(cx)
      {if(cx%10==i)
      y=y*10+i;
      cx=cx/10;
      }
 }
cout<<y;
return 0;
}
```

d) Vom utiliza structura repetitivă „cât timp – execută"

citește x
(x număr întreg)
y←0; i←9
cât timp i≥0 execută
  cx←x
  cât timp cx>0 execută
    dacă cx%10=i atunci
      y← y*10+i
    cx←cx/10
  i←i-1
scrie y

2.

```
cin>>z.pre>>z.pim;
cout<<sqrt(z.pre*z.pre+z.pim*z.pim);
```

3.

```
if(i==1||j==1)A[i][j]=1;
else A[i][j]=(A[i][j-1)+A[i-1][j])%10;
```


Subiectul III

1.

```
#include <iostream>
#include<cstring>
using namespace std;
int i;
char s[201],aux[201];
int main()
{

    cin.get(s,101);
    for(i=0; i<strlen(s)-1; i++)
        if(s[i]==s[i+1])
        { strcpy(aux,s+i+1);
          s[i+1]='#';
          strcpy(s+i+2,aux);
          i++;
        }
    cout<<s;
    return 0;
}
```

2. Vom determina numărul de cifre utilizate pentru scrierea numerelor din intervalul [1, n–1] și a numerelor din intervalul [1, m], funcția va returna diferența celor două valori.

```
int numarare(int n,int m)
{
    int k,p,cn=n,cm=m,i,nrn,nrm,x;
    k=0;p=0;
    do{cn=cn/10;k++;}while(cn);
    do{cm=cm/10;p++;}while(cm);
    nrn=0;x=9;s=9;
    for(i=1;i<k;i++)
    {
        nrn+=i*x;
        s=s+x;x=x*10;
    }
    nrn+=(n-1-s)*k;
    nrm=0;x=9;s=9;
    for(i=1;i<p;i++)
    {
        nrm+=i*x;
        s=s+x;x=x*10;
    }
    nrm+=(m-s)*p;
    return nrm-nrn;
}
```

3. **a)** Vom utiliza un vector de frecvență. Deoarece numerele sunt întregi de două cifre, vom aduna fiecare număr cu 99, astfel încât să lucrăm doar cu valori poztive din intervalul [0,198]. La afișare avem grijă să scădem 99.

b)

```
#include <iostream>
#include<fstream>
using namespace std;
ifstream fin(„bac.txt");
int x,vf[200],i,vfmax;
int main()
{
    while(fin>>x)
    {
        vf[x+99]++;
    }
    for(i=-99;i<=99;i++)
        if(vf[i+99]>vfmax)
            vfmax=vf[i+99];
    for(i=-99;i<=99;i++)
        if(vf[i+99]==vfmax)
            cout<<i<<' ,;
    return 0;
}
```

TEZA 5

Subiectul I.

1. d **2.** d **3.** c **4.** c **5.** c

Subiectul II.

1. **a)** Algoritmul calculează cifra de control a unui număr natural. Se va afișa 3.

b) Pentru orice număr nenul divizibil cu 9 se va afișa 9. Cel mai mic număr este 18, cel mai mare număr este 99.

c)

```
#include <iostream>
using namespace std;
int x,y;
int main()
{
    cin>>x;
    while(x>9)
    {
y=0;
while(x>0)
    {y=y+x%10;
     x=x/10;
    }
 x=y;
}
cout<<y;
return 0;
 }
```

d) Se știe că cifra de control a unui număr se poate determina în funcție de restul împărțirii numărului la 9.

citește x (x număr natural)
dacă x%9=0 atunci
  x←9
altfel
  x←x%9
scrie x

2.

```
per=sqrt((p.v[n].x-p.v[1].x)*
(p.v[n].x-p.v[1].x)+ (p.v[n].y-p.
v[1].y)* (p.v[n].x-p.v[1].x);
for(i=2;i<=n;i++)
   per+=sqrt((p.v[i].x-p.v[i-1].x)*
   (p.v[i].x-p.v[i-1].x)+ (p.v[i].y-
   p.v[i-1].y)* (p.v[i].x-p.v[i-1].x);
```

```
3.  if(i==j||i+j==6)a[i][j]=0;
    else if(i+j<6)
              if(i>j)a[i][j]=4;
              else a[i][j]=1;
         else if(i>j)a[i][j]=3;
              else a[i][j]=2;
```

Subiectul III

```
1.  #include <iostream>
#include<cstring>
using namespace std;
int i;
char s[101],aux[201];
int main()
{

    cin>>s;
    for(i=1; i<=strlen(s); i++)
        { strncpy(aux,s,i);
          aux[i]='\0';
          strcat(aux,s+strlen(s)-i);
          cout<<aux<<' ';
        }
    return 0;
}
```

2. Vom descompune n în factori primi, funcția va returna suma factorilor primi.

```
int divizori(int n)
{
    int s=0,d=2;
    while(n!=1)
    {
        if(n%d==0)
        {
            while(n%d==0)
                n=n/d;
            s=s+d;
        }
        d++;
    }
    return s;
}
```

3. **a)** Vom citi succesiv numerele din fișier, vom determina valoarea cerută chiar la citire (eficiența timp). Determinăm după câți pași va fi eliminat numărul curent. Numărul maxim de pași necesari este numărul de prelucrări. Vom utiliza doar variabile simple (eficiența spațiu de memorie).

b)

```
#include <iostream>
#include<fstream>
using namespace std;
ifstream fin(„bac.txt");
int x,nrp,nrpmax;
int prim(int x)
{
    if(x<=1||x%2==0&&x!=2)return 0;
    for(int i=3; i*i<=x; i=i+2)
        if(x%i==0)return 0;
    return 1;
}
int main()
{
    while(fin>>x)
    {
        nrp=0;
        while(prim(x)==0)
        {
            nrp++;
            x++;
        }
        if(nrp+1>nrpmax)nrpmax=nrp+1;
    }
    cout<<nrpmax;
    return 0;
}
```

TEZA 6

Subiectul I

1. b **2.** b **3.** d **4.** d **5.** b

Subiectul II

1. **a)** Algoritmul determină numărul divizorilor proprii ai unui număr natural. Se va afișa 4.
2. **b)** 2, 3, 4, 5, 7, 9 (numerele prime și pătratele perfecte de numere prime)
3. **c)**

```
#include <iostream>
using namespace std;
int x,y,i;
int main()
{   cin>>x;
    for(i=2;i;=x/2;i++)
        if(x%i==0)y=y+1;
    if(y==0)cout<<1;
    else cout<<y;

 return 0;
}
```

4. **d)** Vom utiliza structura „cât timp – execută”

citește x
(x număr întreg)
y←0
i←2
┌cât timp i ≤ x/2 execută
│ ┌ dacă x% i=0 atunci
│ │ y←y+1
│ └■
│ i←i+1
└■
┌ dacă y=0 atunci
│ scrie 1
│ altfel
│ scrie y
└■


```
2.  A.x==A.y&&B.x==B.y
3.  if(j==1)A[i][j]=i;
    else A[i][j]=A[i][j-1]+5;
```

Subiectul III

```
1.  #include <iostream>
#include<cstring>
using namespace std;
int lmax,nrmax;
char s[101],*p;
int main()
{

    cin.get(s,101);
    p=strtok(s," „);
    while(p)
    {
        if(strlen(p)>lmax)
        {lmax=strlen(p);
        nrmax=1;}
        else if(strlen(p)==lmax)
        nrmax++;
        p=strtok(NULL," „);
    }
    cout<<lmax<<' ,<<nrmax;
    return 0;
}
```

```
2.  int divizori(int n,int a[])
{
    int d,i,x,y,r;
    d=a[1];
    for(i=2; i<=n; i++)
    {
        x=d;
        y=a[i];
        while(y)
        {
            r=x%y;
            x=y;
            y=r;
        }
        d=x;
    }
    return d;
}
```

3. **a)** Vom citi succesiv valorile din fișier. Pentru fiecare valoare citită verificăm dacă este un număr p–compus. În caz afirmativ afișăm primul număr din secvență. Fie a + 1, a + 2,... a + p cele p numere consecutive atunci p*a + p*(p + 1)/2 = x, putem determina în mod unic valoarea a, dacă aceasta există. Algoritmul utilizează doar variabile simple și este liniar.

b)

```
#include <iostream>
#include<fstream>
using namespace std;
ifstream fin(„bac.txt");
int x,p,a;
int main()
{
    fin>>p;
    while(fin>>x)
    {
        if((x-p*(p+1)/2)%p==0)
        {
            a=(x-p*(p+1)/2)/p;
            cout<<a+1<<' ,;
        }
        else cout<<"NU"<<' ,;
    }
    return 0;
}
```

TEZA 7

Subiectul I

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

Subiectul II

1. **a)** Algoritmul determină câtul și restul împărțirii lui x la y. Se vor afișa **9** și **5**.
2. **b)** Orice două valori pentru care câtul împărțirii lui x la y este 9. De exemplu: 64 și 7.
3. **c)**

```
#include <iostream>
using namespace std;
int x,y,k,r;
int main()
{
    cin>>x>>y;
    r=x;
    while(y<=r)
      r=r-y;
    k=(x-r)/y
    cout<<k<<' '<<r;
    return 0;
}
```

4. **d)** citește x, y (x, y numere naturale)

r←x%y
k←(x – r)/y
scrie k, r

```
2.  A.x==0&&B.x==0||A.y==0&&B.y==0
3.  if(i%2==0)A[i][j]=(i-1)*5+j;
    else A[i][j]=(i-1)*5+(6-j);
```

Subiectul III

```
1.  #include <iostream>
#include<cstring>
using namespace std;
int i,k;
char s[101],c[101];
int main()
{

    cin>>s>>c;
    for(i=0;i<strlen(s);i++)
```

```
    {
        k=c[i]-'0';
        if(s[i]+k>'z')s[i]='a'+
        (s[i]+k-'z'-1);
        else s[i]=s[i]+k;
    }
    cout<<s;
    return 0;
}
```

2.
```
void permuta(int n,int a[],int k)
{
    int i,j;
       //inversam primele n-k valori
    for(i=1,j=n-k;i<j;i++,j--)
         swap(a[i],a[j]);
       //inversam elementele de la
       n-k+1 la n
    for(i=n-k+1,j=n;i<j;i++,j--)
         swap(a[i],a[j]);
      // inversam vectorul
    for(i=1,j=n;i<j;i++,j--)
         swap(a[i],a[j]);
}
```

3. **a)** Vom construi un vector de frecvență care memorează cu câte zerouri se termină un număr (exponentul). Afișăm apoi toate puterile lui 10 care au frecvența nenulă. Algoritmul este eficient ca timp de execuție deoarece este liniar, iar ca spațiu de memorie deoarece folosește un vector cu 10 elemente.

b)
```
#include<fstream>
#include<cmath>
using namespace std;
ifstream fin(„bac.in”);
ofstream fout(„bac.out”);
int x,p,i,vf[10];
int main()
{
    while(fin>>x)
    {
        p=log10(x);
        vf[p]++;
    }
    for(i=0; i<=9; i++)
        while(vf[i])
        {
            fout<<pow(10,i)<<' ,;
            vf[i]--;
        }
    return 0;
}
```

TEZA 8

Subiectul I

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

Subiectul II

1. **a)** Algoritmul determină suma divizorilor mai mici decât x. Algoritmul afișează valoarea 1 dacă numărul este perfect (suma divizorilor este egală cu dublul numărului). Pentru 12 se va afișa 16 (1 + 2 + 3 + 4 + 6 = 16)

b) o valoare posibilă pentru x este 6 sau 28.

c)
```
#include <iostream>
using namespace std;
int x,y,i;
int main()
{   cin>>x;
    for(i=1;i;=x/2;i++)
        if(x%i==0)y=y+i;
    if(y==x)cout<<1<<' '<<x;
    else cout<<y;
 return 0;
}
```

d) citește x
(x număr natural, x≠1)
y←0; i←1
dacă i<=x/2 atunci
  repetă
    dacă x% i=0 atunci
      y←y+i
    i←i+1
  până când i>x/2
dacă y=x atunci
  scrie 1 ,' ', x
altfel
  scrie y

2.
```
t.a==t.b&&t.a!=t.c || t.a==t.c&&t.a!=t.
b|| t.c==t.b&&t.a!=t.c
```

3.
```
if(strstr(s,"2019")==s&&strstr
(s+4,"2019")==0)nr++;
```

Subiectul III

1.
```
#include <iostream>
using namespace std;
int a[11][11],n,i,j,k;
int main()
{
    cin>>n;
    for(i=1; i<=n; i++)
        for(j=1; j<=n; j++)
            cin>>a[i][j];
    for(i=1; i<=n; i++)
    {
        k=0;
        for(j=1; j<=n; j++)
            k=k*2+a[i][j];
        cout<<k<<' ,;
    }
    return 0;
}
```


2.
```
int divizor(int n)
{
 int d=2,p=1,k,cn=n;
 //calculam in p produsul factorilor
 primi la putere impara
 d=2;
 while(d*d<=n)
 {
     if(n%d==0)
     {
         k=0;
         while(n%d==0)
         {k++;n=n/d;}
         if(k%2)p=p*d;
     }
     d++;
 }
 if(n>1)p=p*n;
 return cn/p;
}
```

3. **a)** Vom determina pentru fiecare număr printr-un algoritm eficient numărul de valori egale cu 0 de la sfârștul lui x!, acest număr este egal cu puterea lui 5 în descompunerea în factori primi a lui x!. Determinăm maximul de zerouri și numărul de maxime printr-un algoritm clasic. Utilizăm doar variabile simple (complexitate spațiu de memorie), rezolvăm cerința în timpul citirii (complexitate timp).

b)
```
#include <iostream>
#include<fstream>
using namespace std;
ifstream fin(„bac.txt”);
int x,zmax, nrzmax,nrz;
int main()
{
    while(fin>>x)
    {
        nrz=0;
        while(x)
        {
            nrz+=x/5;
            x=x/5;
        }
        if(nrz>nrzmax)
        {
            nrzmax=nrz;
            zmax=1;
        }
        else if(nrz==nrzmax)zmax++;
    }
    cout<<zmax<<' ,<<nrzmax;
    return 0;
}
```

TEZA 9

Subiectul I

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

Subiectul II

1. **a)** algoritmul determină cel mai mare divizor al lui n, diferit de n, se va afișa 13;

b) putem alege orice număr prim de o cifră 2, 3, 5, 7;

c)
```
#include<iostream>
using namespace std;
int a,b,n,c;
int main()
{ cin>>n;
a=1;b=n/2;
while(a!=0&&b>0)
{ c=n;
   while(c>=b)
      c=c-b;
   a=c;
   b=b-1;
}
cout<<b+1;
return 0;
}
```

d) A doua structură while determină restul împărțirii lui n la b.
citește n (n număr natural nenul)
a←1
b←[n / 2]
cât timp a ≠ 0 și b > 0 execută
  c←n%b
  a←c
  b←b-1
b←b+1
scrie b

2.
```
for(i=0;i<M.nrProduse;i++)
if(M.P[i].cant!=0)cout<<M.P[i].cant*
M.P[i].pret<<endl;
else cout<<M.P[i].cod<<endl;
```

3.
```
if (i==j) a[i][j]=6;
else
if (i%2==0)  a[i][j]=5-i;
else a[i][j]=5-j;
```

Subiectul III

1.
```
int circular(int a,int b)
{
 int p=1,ca,n=0;
 ca=a;
 while(ca>9)
 {
     ca=ca/10;
     p=p*10;
     n++;
 }
```

```
    for(int i=0; i<n; i++)
    {
        if(a==b)return i;
        a=(a%10)*p+a/10;
    }
    return -1;
}
```

2.
```
#include <iostream>
#include<cstring>
using namespace std;
int k,l;
char s[256],c[31], *p;
int main()
{
    cin.getline(s,256);
    cin>>c;
    //determinam sufixul lui c
    l=strlen(c)-1;
    while(strchr(„aeiou”,c[l])==0&&
    l>=0)l--;
    strcpy(c,c+l);
    p=strtok(s,” „);
    while(p)
    {
        l=strlen(p)-1;
        while(strchr(„aeiou”,p[l])==0
        && l >=0)l--;
        if(l>=0&&strcmp(c,p+l)==0)
        {
            cout<<p<<endl;
            k++;
        }
        p=strtok(NULL,” „);
    }
    if(k==0)cout<<”NU EXISTA”;
    return 0;
}
```

3. a) Vom citi numerele din fișier succesiv și vom memora într-un vector de frecvență numărul de apariții a fiecărei cifre. Cifrele care apar în palindrom trebuie să apară de un număr par de ori, mai puțin cifra din mijloc. Dacă în fișier există o cifră pară nenulă care apare de cel puțin 2 ori, construim cel mai lung palindrom care începe cu cea mai mare cifră pară, altfel afișăm –1. Algoritmul este eficient din punct de vedere al timpului de execuție deoarece este un algoritm liniar și din punct de vedere al spațiului de memorie deoarece utilizează doar variabile simple și un vector cu 10 elemente.

b)
```
#include <iostream>
#include<fstream>
using namespace std;
ifstream fin(„bac.txt”);
int x,fr[10],exista,i,j,mijloc,p,n;
int main()
{
fin>>n;
for(i=1; i<=n; i++)
{
    fin>>x;
    fr[x]++;
    if(x>0&&x%2==0&&fr[x]>=2)
    exista=1;
}
if(exista)
{
    //determin prima cifra
    for(i=8; i>=2; i=i-2)
        if(fr[i]>=2)
        {
            p=i;
            fr[i]=fr[i]-2;
            break;
        }
    //determin cifra din mijloc
    daca exista, trebuie sa fie
    una singura
     mijloc=-1;
     for(i=9; i>=1; i--)
        if(fr[i]%2==1)
            if(mijloc==-1)
            {
                mijloc=i;
            }
            else
            {
                exista=0;
                break;
            }
     }
     if(exista){
        cout<<p;
        for(i=9; i>=1; i--)
        {
            for(j=1; j<=fr[i]/2;
j++)
                cout<<i;
        }
        if(mijloc!=-1)cout<<mijloc;
        for(i=1; i<=9; i++)
        {
            for(j=1; j<=fr[i]/2;
j++)
                cout<<i;
        }
        cout<<p;
     }
     if(!exista)cout<<-1;
     return 0;
}
```


TEZA 10

Subiectul I

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

Subiectul II

1. a) algoritmul determină cel mai mare divizor a n numere citite succesiv, se va afișa 2;

b) putem alege oricare trei numere prime între ele: 3, 5, 4.

c)
```
#include<iostream>
using namespace std;
int n,i,r,d,x;
int main()
{ cin>>n;
d=0;
for(i=1;i<=n;i++)
{cin>>x;
if(d==0)d=x;
else
do{r=x%d;x=d;d=r;}while(r);
d=x;
}
cout<<d;
return 0;
}
```

d) citește n (n număr natural)

```
d←0
pentru i←1, n execută
    citește x
    dacă d=0 atunci
        d←x
    altfel
        r←x%d
        x← d
        d←r
        cât timp r>0 execută
            r←x%d
            x← d
            d←r
        d←x
scrie d
```

2.
```
 cin>>E.nrJucatori;
for(i=0;i<E.nrJucatori;i++)
   cin>>E.juc[i]. nrTricou>>
   E.juc[i].marimeTricou;
```

3.
```
if (i<3) {
a[i][j]=(i+j)%6;
a[i][5-j]=(i+j)%6;
}
else {
   a[i][j]=a[5-i][j];
   a[i][5-j]=a[5-i][j];
}
```

Subiectul III

1. Cel mai mic divizor diferit de 1 a lui n este prim, vom căuta valoarea divizorului în vectorul v prin căutare binară. Dacă nu îl găsim îl vom insera în v pe poziția furnizată de căutarea binară.

```
void divprim(int n,int v[],int &k)
{
    int d,ls,ld,mij,ok=0;
    d=n;
    for(int i=2;i*i<=n;i++)
    if(n%i==0){d=i;break;}
    ls=1;ld=k;
    while(ls<=ld&&!ok)
    {
        mij=(ls+ld)/2;
        if(v[mij]==d)ok=1;
        else if(d<v[mij])ld=mij-1;
        else ls=mij+1;
    }
    if(!ok)
    {
        for(int i=k;i>=ls;i--)
            v[i+1]=v[i];
        v[ls]=d;k++;
    }
}
```

2.
```
#include <iostream>
#include<cstring>
using namespace std;
int l,k;
char s[256],rez[256],c[256], *p;
int main()
{
    cin.getline(s,256);
    p=strtok(s,” “);
    while(p)
    {
        l=strlen(p);
        if(l%2==0)
        {

   strncpy(c,p,l/2);c[l/2]='\0';
            if(strcmp(c,p+l/2)==0)
                {strcpy(p+l/2,”*”);
                k++;}
        }
        strcat(rez,p);strcat(rez,” “);
        p=strtok(NULL,” “);
    }
    strcpy(s,rez);s[strlen(s)-1]='\0';
    if(k)cout<<s;
    else cout<<”NEMODIFICAT”;
    return 0;
}
```

3. **a)** Vom citi numerele din fișier succesiv și vom determina pentru fiecare număr citit x, cea mai

mare putere a lui 2 mai mică sau egală cu x, vom memora într-un vector de frecvență numărul de numere citite pentru care puterea lui 2 este aceeași. Deoarece valorile din fișier sunt de tip int, care se memorează pe 32 de biți, vectorul de frecvență va avea 32 de elemente. Algoritmul este eficient din puncte de vedere al timpului de execuție deoarece este un algoritm liniar și din punct de vedere al spațiului de memorie deoarece utilizează doar variabile simple și un vector cu 32 elemente.

b)
```
#include <iostream>
#include<fstream>
using namespace std;
ifstream fin(„bac.txt”);
int x,fr[32],p,k,frmax,i;
int main()
{
    while(fin>>x)
    {
        p=1;
        k=0;
        while(p*2<=x)
        {
            p=p*2;
            k++;
        }
        fr[k]++;
    }
    for(i=0;i<32;i++)
        if(fr[i]>=frmax){frmax=fr[i];
        k=i;}
    cout<<k;
    return 0;
}
```

TEZA 11

Subiectul I

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

Subiectul II

1. **a)** algoritmul determină câte perechi de divizori au aceeași paritate; se va afișa 2.

b) cel mai mic număr cu această proprietate este 102 (are perechile de divizori 1 cu 102 și 2 cu 51, 3 și 34, 6 și 17 elementele unei perechi sunt de parități diferite)

c)
```
#include<iostream>
using namespace std;
int a,b,n,k;
int main()
{
    cin>>n;
    a=1;
    k=0;
    while(a*a<=n)
    {
        if(n%a==0)
        {
            b=n/a;
            if(a%2==b%2)k++;
        }
        a=a+1;
    }
    cout<<k;
    return 0;
}
```

d) citește n (n număr natural nenul)

k←0

a←1

dacă a * a ≤ n atunci
  repetă
    dacă n % a=0 atunci
      b←[n /a]
      dacă a % 2=b % 2 atunci
        k←k + 1
  a←a+1
  până când a*a>n

scrie k

2.
```
for(i=0;i<D.nrFisiere;i++)
    sum+=D.F[i].dimensiune;
```

3. Se va afișa șirul: bac la informatica

Subiectul III

1.
```
#include <iostream>
using namespace std;
int a[50][50],v[50],n,i,j,k;
int main()
{
    cin>>n;
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            cin>>a[i][j];
    for(j=0; j<n; j++)
    {
        k=0;
        for(i=0; i<n; i++)
            k+=a[i][j];
        v[j]=k;
    }
    for(i=0; i<n; i++)
    {
        k=0;
        for(j=0; j<n; j++)
            k+=a[i][j];
        if(k==0)
            for(j=0; j<n; j++)
                a[i][j]=v[j];
    }
    for(i=0; i<n; i++)
    {
        for(j=0; j<n; j++)
            cout<<a[i][j]<<' ';
        cout<<endl;
    }
    return 0;
}
```

2.
```
void nPrime(int n,int &k, int p[])
{
    int i, a, b, r;
    for(i=1;i<n;i++)
    {
        a=i;b=n;
        while(b)
        {r=a%b;
        a=b;b=r;
        }
        if(a==1)
        {k++;
        p[k]=i;}
    }
}
```

3. **a)** Putem stabili următoarele formule de recurență: Nrstive(1) = 1, Nrstive(n) = 2*Nrstive (n–1) + 1, Nrcutii(1) = 1, Nrcutii(n) = 2*Nrcutii (n) + n. Pornind de la aceste formule de recurență se poate construi un algoritm liniar care determină valorile cerute, utilizând numai variabile simple.

b)
```
#include <iostream>
#include<fstream>
using namespace std;
ifstream fin(„bac.txt”);
int n,i,Nrstive, Nrcutii;
int main()
{
    fin>>n;
     Nrstive=1;Nrcutii=1;
    for(i=2;i<=n;i++)
    {   Nrstive=2*Nrstive+1;
        Nrcutii=2*Nrcutii+i;
    }
     cout<<Nrstive<<' '<<Nrcutii;
    return 0;
}
```

TEZA 12

Subiectul I

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

Subiectul II

1. **a)** algoritmul afișează în ordine descrescătoare toate numerele din intervalul [a, b] cu proprietatea că în reprezentarea în baza 2 numărul de cifre 1 este egal cu numărul de cifre egale cu 0; se va afișa 12 10 9 *;

b) cel mai mare număr este 34;

c)
```
#include<iostream>
using namespace std;
int a,b,x,y,k;
int main()
{
    cin>>a>>b;
    for(x=b;x>=a;x--)
    {
        y=x;k=0;
        while(y>0)
        {
            k=k+1-2*(y%2);
            y=y/2;
        }
        if(k==0)
            cout<<x<<' ';
    }
    cout<<'*';
    return 0;
}
```

d) citește a, b (a, b numere naturale, a≤b)

y←0

pentru x←b, a, –1 execută
  y←x
  k←0
  dacă y>0 atunci
    repetă
      k←k+1–2*(y%2)
      y←[y/2]
    până când y=0
  dacă k=0 atunci
    scrie x, ' '

scrie '*'

1.
```
for(i=0;i<100;i++)
if(T[i].lungime==T[i].latime&&
T[i].lungime==T[i].inaltime)
cout<<T[i].culoare;
```

2.
```
if(i%3+j%3==4) a[i][j]=6;
else a[i][j]=i%3+j%3;
```

Subiectul III

1.
```
#include <iostream>
#include<cstring>
using namespace std;
int i,lg,etape,ok;
char s[251];
int main()
{
    cin>>s;
    i=0;
    while(i<strlen(s))
    {
        lg=i;
        while(s[lg]==s[lg+1]&&
        lg+1<strlen(s))
            lg++;
        if(lg!=i)
```

```
        {
            strcpy(s+i,s+lg+1);
            i--;
            if(ok==0)etape++;
            ok=1;
        }
        else {i++;ok=0;}
    }
    cout<<s<<' ,<<etape;
    return 0;
}
```

2.
```
int perechi(int n, int m)
{
    int i, j, a, b, r,k=0;
    for(i=1;i<n;i++)
    {
        for(j=i+1;j<=n;j++)
        {
            a=i;b=j;
            while(b)
            { r=a%b;a=b;b=r;}
            if(i*j/a==m)k++;
        }
    }
    return k;
}
```

3. **a)** Vom citi numerele din fișier succesiv și vom memora într-un vector de frecvență numărul de apariții a fiecărui număr, deoarece valorile din fișier pot fi negative, vom aduna la fiecare valoare citită pe 9, astfel încât valorile din fișier să se afle în intervalul [0, 18]. Algoritmul este eficient din puncte de vedere al timpului de execuție deoarece este un algoritm liniar și din punct de vedere al spațiului de memorie deoarece utilizează doar variabile simple și un vector cu 20 elemente.

b)
```
#include <iostream>
#include<fstream>
using namespace std;
ifstream fin(„bac.txt”);
int x, fr[20], i, frmax;
int main()
{
    while(fin>>x)
        fr[x+9]++;
    for(i=0;i<=18;i++)
        if(fr[i]>frmax)frmax=fr[i];
    for(i=0;i<=18;i++)
        if(fr[i]==frmax)cout<<i-9<<' ,;
    return 0;
}
```

TEZA 13

Subiectul I

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

Subiectul II

1. **a)** algoritmul afișează al n-lea termen a șirului lui Fibonacci. Se va afișa 13.

b) cel mai mic număr este 12 (se va afișa 144);

c)
```
#include<iostream>
using namespace std;
int n,x,y,m,z;
int main()
{
    cin>>n;
    x=0;y=1;m=n/2;
    while(m>0)
    {
        x=x+y;
        y=y+x;
        m--;
    },
    z=(1-n%2)*x+n%2*y;
    cout<<z;
    return 0;
}
```

d) citește n (n număr natural)

x←0; y←1;
m←[n/2];
dacă m>0 atunci
  repetă
    x←x+y
    y←y+x
    m←m-1
  până când m = 0
z ← (1 – n%2)*x + (n%2)*y
scrie z

2.
```
C.centru.y==0&&abs(C.centru.x)<=
C.raza
```

3.
```
if(i==5||j==1) a[i][j]=5;
else a[i][j]=a[i-1][j+1]%2+a[i]
[j-1]%2;
```

Subiectul III

1.
```
#include <iostream>
#include<cstring>
using namespace std;
int i;
char s[201],rez[201],*p;
int main()
{
    cin.get(s,201);
    p=strtok(s,” „);
    while(p)
    {
        //verific daca p are cifre
        int ok=0;
        for(i=0; i<strlen(p)&&!ok; i++)
            if(strchr(„0123456789”,
            p[i])) ok=1;
        if(!ok)
        {
            if(p[0]>='a'&&p[0]<='z')
                p[0]=p[0]-32;
            strncat(rez,p,1);
        }
        p=strtok(NULL,” „);
    }
    cout<<rez;
    return 0;
}
```

2.
```
int rotund(int n)
{
    if(n%2)return 0;
    int a=n%2,b;n=n/2;
    while(n)
    {
        b=n%2;n=n/2;
        if(a==b)return 0;
        a=b;
    }
    return 1;
}
```

3. **a)** Citim x și determinăm în variabila p valoarea 10^k, unde k este numărul de cifre al lui x. Se poate observa că variabila x este sufix al lui y dacă y%p==x. Memorăm la fiecare pas numărul curent cu această proprietate. Algoritmul este eficient din punct de vedere al timpului de execuție deoarece este un algoritm liniar și din punct de vedere al spațiului de memorie deoarece utilizează doar variabile simple.

b)
```
#include <iostream>
#include<fstream>
using namespace std;
ifstream fin(“bac.txt”);
int x, y, u, cx, p;
int main()
{
    fin>>x;
    p=1;cx=x;
    while(cx)
    {p=p*10;cx=cx/10;}
    while(fin>>y)
        if(y%p==x)u=y;
    if(u!=-1)cout<<u;
    else cout<<”NU EXISTA”;
       return 0;
}
```

TEZA 14

Subiectul I.

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

Subiectul II.

1. **a)** algoritmul verifică dacă valorile negative citite sunt în ordine strict descrescătoare. Se va afișa 1.

b) un exemplu poate fi: –2, –8, –4, 1.

c)
```
#include<iostream>
using namespace std;
int n,x, k,p;
int main()
{
    cin>>n;
    k=1;
    p=0;
    while(n>0)
    {
        cin>>x;
        if(x<0&&p>x)
            p=x;
        else if(x<0&&p<=x)
            k=0;
        n--;
    }
    cout<<k;
    return 0;
}
```

d) citește n (n număr natural nenul)

k←1; p←0
pentru i←n, 1, –1 execută
  citește x (x, număr întreg)
  dacă x < 0 și p > x atunci
    p←x
  altfel
    dacă x<0 && p<=x atunci
      k←0
scrie k

2.
```
struct Produs{
int cod;
struct {int luna, an;}data_expirarii;
float pret_furnizor[5];
}P[100];
```

3.
```
if (j==0 || j==6 ) a[i][j]='*';
else if ((i<=3) && (j+i==6 || i==j))
a[i][j]='*';
     else  a[i][j]='-';
```

Subiectul III

1.
```
#include <iostream>
#include<cstring>
using namespace std;
int i,n,l,paragraf;
char s[201],*p,sir[201][201];
int main()
{
    cin>>n;
    cin.get();
    cin.get(s,201);
    p=strtok(s,” „);
    l=0;
    while(p)
    {

        if(strlen(p)>n)
        {
            cout<<”IMPOSIBIL”;
```

```
        return 0;
    }
    else
    if(l==0)
    {
        l=strlen(p);
        strcat(sir[paragraf],p);
    }
    else
    {
        if(l+strlen(p)+1<=n)
        {
            l=l+strlen(p)+1;
            strcat(sir[paragraf],
            " „);
            strcat(sir[paragraf],p);
        }
        else
        {
            for(i=l+1; i<n; i++)
                strcat(sir
                [paragraf],"*");
            l=strlen(p);
            paragraf++;
            strcat(sir[paragraf],p);
        }
    }
        p=strtok(NULL," „);
    }
    cout<<paragraf+1<<endl;
    for(i=0;i<=paragraf;i++)
        cout<<sir[i]<<endl;
    return 0;
}
```

2.
```
int regulat(int n)
{
    //generez numerele regulate pana la n
    int i2=1,i3=1,i5=1, v[1000],x=1,k=1;
    v[1]=1;
    while(x<n)
    {
        int v2=v[i2]*2;
        int v3=v[i3]*3;
        int v5=v[i5]*5;
        x=min(min(v2,v3),v5);
        if(x==v2)i2++;
        if(x==v3)i3++;
        if(x==v5)i5++;
        k++;v[k]=x;
        if(v[k]>n)return v[k-1];
    }
}
```

3. **a)** Citim k și apoi succesiv valorile din fișier, calculând pentru fiecare frecvența de apariție. Parcurgem descrescător numerele de la 99 la 0, adunând frecvențele, când suma este cel puțin k, ultimul număr pentru care am adunat frecvența este numărul căutat. Dacă în fișier sunt mai puțin de k valori se va afișa –1. Algoritmul este eficient din punct de vedere al timpului de execuție deoarece este un algoritm liniar și din cel al spațiului de memorie deoarece utilizează doar variabile simple și un vector de frecvență cu 100 de elemente.

b)
```
#include <iostream>
#include<fstream>
using namespace std;
ifstream fin("bac.txt");
int x,k,fr[100],i,s;
int main()
{
    fin>>k;
    while(fin>>x)
        fr[x]++;

    for(i=99; i>=0; i--)
    {
        s=s+fr[i];
        if(s>=k)
        {
            cout<<i;
            return 0;
        }
    }
    cout<<-1;
    return 0;
}
```

TEZA 15

Subiectul I

1. d 2. c 3. b 4. b 5. d

Subiectul II

1. **a)** algoritmul afișează produsul factorilor primi care apar la putere impară. Se va afișa 231.

b) cel mai mic număr este 25, cel mai mare este 81.

c)
```
#include<iostream>
using namespace std;
int n,p,m,x;
int main()
{
    cin>>n;
    p=1;
    m=0;
    x=2;
    while(n>=x)
    {
        if(n%x==0)
        {
            m++;
            n=n/x;
        }
        else
        {
            if(m%2!=0)
                p=p*x;
            m=0;
            x++;
        }
    }
    if(m%2!=0)p=p*x;
    cout<<p;
    return 0;
}
```

d) citește n (n număr natural, n>1)

p←1; m←0; x←2
- dacă n >= x atunci
  - repetă
    - dacă n%x=0 atunci
      - m←m + 1; n←[n/x]
    - altfel
      - dacă m%2 ≠ 0 atunci
        - p←p*x
      - m←0; x←x+1
  - până când n<x
- dacă m%2 ≠ 0 atunci
  - p←p*x

scrie p

2.
```
if(candidat.mat>=5&&candidat.
mat>=5&&candidat.rom>=5)
if((float(candidat.mat+candidat.
mat+candidat.rom)>=6)
candidat.rez='A';else candidat.
rez='R';
```

3.
```
{
 a[i][j]=i<j?i:j;
 a[7-i][j]=a[i][7-j]=a[7-j]
 [7-i]=a[i][j];
}
```

Subiectul III

1.
```
#include <iostream>
#include<cstring>
using namespace std;
int i,j,n,ok;
char s[201],sn[201];
int main()
{
    cin>>s;
    ok=1;
    for(i=0; i<strlen(s)/2&&ok; i++)
            if(s[i]!=s[str-
len(s)-i-1])ok=0;
    if(ok)
        {   cout<<s;
            return 0;
        }
    for(i=0; i<strlen(s)-1; i++)
    {
        //adaugam i+1 caractere la
        sfarsitul lui s
        //verificam daca este
        palindrom
        strcpy(sn,s);
        for(j=i;j>=0;j--)
            strncat(sn,s+j,1);
        ok=1;
        for(j=0; j<strlen(sn)/2&&ok;
        j++)
            if(sn[j]!=sn[strlen(sn)-
            j-1])ok=0;
        if(ok)
        {
            strcpy(s,sn);
            cout<<s;
            return 0;
        }
    }
}
```

2.
```
void compact(int &n, int v[])
{
    int i,j;
    for(i=1; i<n;)
        if(v[i]==v[i+1])
        {
            for(j=i+2; j<=n; j++)
                v[j-1]=v[j];
            n--;
        }
        else i++;
}
```

3. **a)** Vom citi numerele din fișier succesiv și vom adăuga valoarea citită la secvența curentă, sau începem o nouă secvență dacă nu sunt satisfăcute condițiile curente. Algoritmul este eficient din punct de vedere al timpului de execuție deoarece este un algoritm liniar și din cel al spațiului de memorie deoarece utilizează doar variabile simple.

b)
```
#include <iostream>
#include<fstream>
using namespace std;
ifstream fin(„bac.txt");
int x,y,vf,lg,lgmax,nr,i,n;;
int main()
{   fin>>n;
    fin>>x;
    lg=1;
    for(i=2;i<=n;i++)
    {fin>>y;
        if(vf==1&&x<y)lg++;
        else if(vf==0&&x>y)lg++;
        else if(vf==0&&x<y)
```

```
        {
            lg++;
            vf=1;
        }
        else
        {
            if(vf==1&&lg>lgmax)
            lgmax=lg,nr=1;
            else if(vf==1&&lg==lgmax)
            nr++;
            lg=1;vf=0;
        }
        x=y;
    }
    if(vf==1&&lg>lgmax)lgmax=lg,nr=1;
    else if(vf==1&&lg==lgmax)nr++;
    cout<<lgmax<<' ,<<nr;
    return 0;
}
```

Teza 16

Subiectul I

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

Subiectul II

1. **a)** algoritmul determină perechi de numere citite succesiv au proprietatea că primul număr din pereche are cel mult 2 cifre și al doilea număr din pereche are cel puțin 2 cifre; se va afișa 2.

b) putem alege 4 numere de o cifră 2, 3, 5, 7.

c)
```
#include<iostream>
using namespace std;
int i,n,x,y,k;
int main()
{ cin>>n>>x;
for(i=1;i<=n-1;i++)
{ cin>>y;
     if(x<=99&&y>=10)
         k++;
     x=y;
}
cout<<k;
return 0;
}
```

d) citește n, x (n, x numere naturale)

i←1

┌cât timp i ≤ n–1 execută

│ citește y (x, y numere întregi, x ≤ y)

│ ┌dacă x ≤ 99 și y ≥ 10 atunci

│ │ k←k+1

│ └■

│ x←y

│ i← i +1

└■

scrie k

2.
```
struct Campanie{
char titlu[31];
struct {int luna, an;}start;
int durata,procent;
}C[51];
```

3.
```
cin>>s;
n=strlen(s);
for(i=0;i<n;i++)
{
     if (strstr(s+i, "info"))
     cout << s+i << '\n';
}
```

Subiectul III

1. Rezolvăm problema de la citire, construim 2 vectori l și c care memorează câte valori egale cu 'X' sunt pe fiecare linie, coloană, iar pentru cele 2 diagonale utilizăm 2 variabile simple.

```
#include <iostream>
using namespace std;
int l[51],c[51],n,i,j,dp,ds,ok;
char a[51][51];
int main()
{
    cin>>n;
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
        {
            cin>>a[i][j];
            if(a[i][j]=='X')
            {
                if(i==j)dp++;
                if(i+j==n-1)ds++;
                l[i]++;
                c[j]++;
            }
        }
    if(dp==n||ds==n)ok=1;
    for(i=0;i<n;i++)
        if(l[i]==n||c[i]==n)ok=1;
    if(ok)cout<<"DA";
    else cout<<"NU";
    return 0;
}
```

2. Parcurgem cifrele lui n de la stânga la dreapta. Când găsim o cifră mai mică decât cifra dată o inserăm.

```
void nMax(int &n,int c)
{
    int cn=n, p=1,cif,ok=0;
    while(n>9)
    {n=n/10;p=p*10;}
    n=0;
    while(p)
    {
        cif=(cn/p)%10;p=p/10;
        if(c>cif&&!ok)
        {n=n*10+c;ok=1;}
        n=n*10+cif;
    }
    if(!ok)n=n*10+c;
}
```

3. **a)** Vom citi numerele din fișier succesiv și vom determina intervalele asociate, pentru fiecare interval asociat curent determinăm intersecția cu intervalul intersectie de până atunci. Dacă intervalul curent este {a, b}, atunci determinăm maximul valorilor a și minimul valorilor b. Dacă Maxa este mai mare decât Minb, afișăm mulțimea vidă. Algoritmul este eficient din punct de vedere al timpului de execuție deoarece este un algoritm liniar și din punct de vedere al spațiului de memorie deoarece utilizează doar variabile simple.

b)
```
#include <iostream>
#include<fstream>
using namespace std;
ifstream fin(„bac.txt");
int x,y,a,b,maxa,minb,ok;
int main()
{
    fin>>x;
    a=x;
    ok=0;
    while(fin>>y)
    {
        if(y==x+1)b=y;
        else
        {
            if(ok==0)maxa=a,minb=b,ok=1;
            else
            {
                if(maxa<a)maxa=a;
                if(minb>b)minb=b;
            }
            a=y;
        }
        x=y;
    }
    if(ok==0)maxa=a,minb=b,ok=1;
    else
    {
        if(maxa<a)maxa=a;
        if(minb>b)minb=b;
    }
    if(minb>=maxa)cout<<maxa<<' ,<<minb;
    else cout<<"multimea vida";
    return 0;
}
```

TEZA 17

I 1. b **I 2.** c **I 3.** c **I 4.** b **I 5.** b

II1.
```
if (p.dexp.an>azi.an|| (p.dexp.an==
azi.an && azi.luna<p.dexp.luna)
|| p.dexp.an==azi.an && azi.luna<p.
dexp.luna&& azi.zi<p.exp.zi)
  cout<<"produsul nu este expirat";
else
 cout<<"produsul este expirat";
```

II2.
```
for(i=0;i<5;i++)
  for(j=0;j<5;j++)
   if(i==j)
    A[i][j]='A';
   else
    if(i<j)
     A[i][j]=A[i][j-1]+1;
    else
     if(j==0)
      A[i][j]=A[i-1][4];
     else
      A[i][j]=A[i][j-1]+1;
```

II 3. a) 2664 **II3 b.** 2 0 500 500 sau 4 3 334 103 533 133 (orice set de n numere, pentru care suma numerelor care conțin cifra y de un număr par de ori este egală cu 1000).

II 3. c)
```
#include <iostream>
using namespace std;
int main()
{   int e,n,y,i,x,cx,nr;
    cin>>n>>y;
    e=0;
    for(i=1;i<=n;i++)
    {
        cin>>x;
        cx=x;
        nr=0;
        while(x!=0)
        {
            if(x%10==y)
               nr++;
               x=x/10;
        }

        if(nr%2==0 && nr>0)
           e=e+cx;
    }
    cout<<e;
return 0;
}
```

II 3. d)

citește n, y (y număr natural, y<10, n număr natural nenul)

```
e←0; i←1
┌cât timp i<=n execută
  citește x (număr natural nenul)
  cx←x
  nr←0
  ┌cât timp x≠0 execută
  │ ┌dacă x%10=y
  │ │  atunci
  │ │   nr←nr + 1
  │ └
  │ x←[x/10]
  └
  ┌dacă nr%2=0 și nr>0
  │ atunci
  │ e←e+cx
  └
  i←i+1
└
scrie e
```

III 1.

```
#include <iostream>
#include<cstring>
using namespace std;

int main()
{char s[251],voc[]="aeiouAEIOU",aux;
int i;
cin.get(s,251);
for(i=0;i<strlen(s)-1;i++)
    if((strchr(voc,s[i])&&!strchr
    (voc,s[i+1])&&
s[i+1]!=' ,)||(!strchr(voc,s[i])&&
s[i]!=' , && strchr(voc,s[i+1])))
{
    aux=s[i];s[i]=s[i+1];s[i+1]=aux;
}
    cout <<s;
    return 0;
}
```

III 2.

```
int termen(int n)
{   int i,j,k;
    i=1;k=3;
    while(i<=n)
          {   j=1;
              while(j<=k&& i<=n)
                {
                if(i==n)
                return j;
                    j++;i++;
                }
                j=k-1;
                while(j>1&& i<=n)
                {
                if(i==n)
                return j;
                    j--;i++;
                }
          k+=3;
        }
}
```

III 3. a) Pentru fiecare valoare a ultimei cifre **u** a unui număr din șirul citit se reține numărul de ordine din fișier al primului număr ce se încheie cu cifra prin **prord[c]**, numărul de ordine al ultimului număr din fișier ce se încheie cu cifra u prin **ultord[c]** și lungimea unei secvențe de tipul cerut prin **lg[c], lg[c]=ultord[c]-prord[c]+1.** Programul este eficient ca timp de execuție deoarece la citirea fiecărui număr din fișier se completează vectorii prord, ultord pentru ultima cifră a numărului. Se parcurg vectorii prord, ultord pentru fiecare cifră, se calculează lungimea secvenței corespunzătoare cifrei și se determină lungimea maximă a unei secvențe și poziția ultimei secvențe din fișier dacă există mai multe secvențe de lungime maximă.

III3. b)

```
#include <iostream>
#include<fstream>
using namespace std;
ifstream fin(„bac.txt");
int prord[10],lg[10],ultord[10];
int main()
{int n;
 int x,uc,nrord=0,lgmax=0,i,poz;
 while(fin>>x)
 {  nrord++;
    uc=x%10;
    if( lg[uc]==0)
      {prord[uc]=ultord[uc]=nrord;
      lg[uc]=1;}
    else
        ultord[uc]=nrord;
}
for(i=0;i<=9;i++)
{ cout<<prord[i]<<"
   „<<ultord[i]<<endl;
    if(lg[i]!=0)
    {
     lg[i]=ultord[i]-prord[i]+1;
     if(lg[i]>lgmax)
     {
         lgmax=lg[i];
         poz=i;
     }
     else
      if(lg[i]==lgmax)
         if (poz<prord[i])
            poz=i;
    }
}
if(lgmax==1)
    cout<<"Nu exista";
```



```
else
    cout<<lgmax<<" „<<prord[poz];

    return 0;
}
```

TEZA 18

I 1. c **I 2.** b **I 3.** c **I 4.** d **I 5** .b

II 1.

```
 int i,num_min,nr=0;
num_min=x[0].numitor;
for(i=1;i<20;i++)
  if(x[i].numitor<num_min)
{num_min= x[i].numitor;
  nr=1;
}
 else
  if(x[i].numitor==num_min)
     nr++;
cout<<nr;
```

II 2.

```
for(i=0;i<5;i++)
  for(j=0;j<5;j++)
   if(i==0)
     A[i][j]=2;
    else
     if(j==0)
      A[i][j]=2;
     else
      A[i][j]=A[i][j-1]+A[i-1][j];
```

II 3 a) 35

II 3 b) un șir de numere prime încheiat cu 0, de exemplu: 3 5 7 0.

II 3 c)

```
#include <iostream>
using namespace std;
int main()
{ int m,t,x,d,nr,a;
  cin>>a;
  m=0;t=0;
  do
  {x=a;d=2;nr=0;
   while(a>1)
   {
    if(a%d==0)
    {
      while(a%d==0)
         a=a/d;
      nr++;
    }
    d++;
   }
  if(nr>=m)
  {
      m=nr;
      t=x;
  }
  cin>>a;
}
while(a!=0);
cout<<t;
return 0;
}
```

II 3. d)

```
citește a (a număr natural nenul)
 m←0
 t←0
┌cât timp a≠0 execută
   x←a
   d←2
   nr←0
  ┌cât timp a>1 execută
  │  ┌dacă a%d=0
  │  │ atunci
  │  │  ┌cât timp a%d=0 execută
  │  │  │   a=[a/d]
  │  │  └
  │  │ nr ←nr+1
  │  └
  │  d ←d+1
  └
  ┌dacă nr>=m
  │  atunci
  │    m←nr
  │    t←x
  └
 citește a
└
  scrie t
```

III 1.

```
#include <iostream>
#include<cstring>
using namespace std;

int main()
{  char c[21],t[201],prop_
   max[201]="",*p,prop[201],
   cprop[201],*cuv;
    int nr,nrmax=0;
    cin.getline(c,21);cin.
   getline(t,201);
    p=strchr(t,'.');
    while(p)       //separare
   propozitii
    {   nr=0;
        strncpy(prop,t,p-t+1);  //
   extrag o propozitie
        prop[p-t+1]=NULL;
        strcpy(cprop,prop);  //
   copiez propozitia in alt sir
        cuv=strtok(cprop," „);
        while(cuv)     //extrag
   cuvinte din propozitie
        {  if(strcmp(cuv,c)==0)
```

```
            nr++;
        cuv=strtok(NULL," „);
    }
    if(nr>nrmax)
    {   nrmax=nr;
        strcpy(prop_max,prop);
    }
    strcpy(t,p+1);  //elimin
propozitia prelucrata din text
    p=strchr(t,'.');
  }
cout<<prop_max;
return 0;
}
```

III 2.

```
int perechi(int n, int v[])
{    int nr=0,i,j,ogl,c;
     for(i=1;i<n;i++)
     for(j=i+1;j<=n;j++)
     {  c=v[j];ogl=0;
        while(c)
        {   ogl=ogl*10+c%10;
            c/=10;
         }
        if(v[i]==ogl)
            nr++;
return nr;
}
```

III 3. a) La citirea numerelor din fișier se determină frecvența fiecărei cifre ce apare în numerele citite, prin vectorul f (program eficient ca timp de execuție). Se parcurge vectorul f, de la cifra 9 la cifra 2 și se memorează cifra pentru care perechea (cifra, cifra – 2) formează un număr de valoare maximă. Dacă notăm această cifră cu c, numărul afișat va fi – 2. Programul este eficient din punct de vedere al memoriei utilizate deoarece utilizează un număr minim de variabile.

III 3. b)

```
#include <iostream>
using namespace std;
int f[10];  //frecventa cifrelor
pentru numerele din fisier
int main()
{int n,cifra,nrcifre=0,s=0,c;
 while(fin>>n)
 {   while(n)
     {   f[n%10]++;
         n/=10;
     }
}
for(c=9;c>=2;c--)
     if(f[c]!=0&&f[c-2]!=0)
{    s=f[c]+f[c-2];
     if(s>nrcifre)
     {   nrcifre=s;
         cifra=c;
     }
}
if(s==0)
    cout<<"Nu exista";
else
{
    for(c=1;c<=f[cifra];c++)
        cout<<cifra;
    for(c=1;c<=f[cifra-2];c++)
        cout<<cifra-2;
}
    return 0;
}
```

TEZA 19

I 1. b **I 2.** a **I 3.** b **I 4.** d) $4^* A_{24}^3 = 48756$; **I 5.** c

II 1.

```
int lg_max=0;
for(int i=0;i<20;i++)
  if(strlen(c[i].cuv)>lg_max&&c[i].
  frecv>2)
    lg_max=strlen(c[i].cuv);
cout<<lg_max;
```

II 2.

```
for(i=0;i<4;i++)
  for(j=0;j<4;j++)
     if(i%2==0)
    if(j<2)
      A[i][j]='a';
      else
      A[i][j]='b';
    else
     if(j<2)
      A[i][j]='A';
      else
      A[i][j]='B';
```

II 3. a) k = 49 și q = 35 **II 3. b)** a = 2 și b = 3

II 3 c)

```
#include <iostream>
using namespace std;
int main()
{   int a,b,k,q,x,y;
    k=0;q=0;
    cin>>a>>b;
    for(x=1;x<=a;x++)
        for(y=1;y<=b;y++)
         if(y%x==0)
          k+=y;
         else
          q+=y;
    cout<<k<<" „<<q;
    return 0;
}
```


II 3. d)

citește a,b (a≤ b, a, b numere naturale)
k←0; q←0;x←1
cât timp x<=a execută
  pentru y←1, b execută
    dacă y%x=0
     atunci
       k←k+y
     altfel
       q←q+y
x←x+1
scrie k," ",q

III 1.

```
#include <iostream>
#include<cstring>
using namespace std;
int main()
{char s1[41],s2[41],voc[]="aeiou",
aux[41]; int i,p=0;
 cin>>s1>>s2;
 for(i=0;i<strlen(s1);i++)
     if(strchr(voc,s1[i])!=0)
         {// inserare vocala din
         sirul s1 pe pozitia para p
         din s2
           strcpy(aux,s2+p);
           s2[p]=s1[i];s2[p+1]=NULL;
           strcat(s2,aux);
           p=p+2;
           strcpy(s1+i,s1+i+1);//
           eliminare vocala din
           sirul s1
           i--;
         }
 cout<<s1<<endl<<s2;
 return 0;
}
```

III 2.

```
void factor(int n,int v[],int &fp,
int&p)
{int f[100],pt[100],i,c,nf=0,d,pf,
poz,ok=0;
//nf- nr factori,f-factori primi,
p-puteri factori primi
    fp=1;
    c=v[1];
    d=2;
//determin factorii primi si puterile
lor pt v[1] si ii memorez in //
vectorul f, puterile lor le memorez
in vectorul pt
    while(c>1)
    {pf=0;
        while(c%d==0)
        {
            c/=d;
            pf++;
        }
        if(pf!=0)
        {
            nf++;
            f[nf]=d;
            pt[nf]=pf;
        }
        d++;
}
poz=1;
int j;
for(i=2;i<=n&&fp>0;i++)
{c=v[i];ok=0;
// f[j] este factor prim comun
al lui v[1] si v[i]
 for(j=poz;j<=nf;j++)
     if(c%f[j]==0)
   { pf=0;ok=1;
     while(c%f[j]==0)
     {pf++;
      c/=f[j];
     }
     if(pf<pt[j])
        pt[j]=pf;
      fp=f[j];p=pt[j];poz=j;
     break;
     }
    if(ok==0)
    {
        fp=-1;p=0;
    }
}
```

III 3. a) Programul este eficient ca timp de execuție deoarece generează direct numerele care trebuie afișate, alegând în ordine crescătoare primele 3 cifre impare $c_1c_2c_3$, palindromul afișat va fi $c_1c_2c_3\ c_3\ c_2\ c_1$. Programul este eficient din punct de vedere al memoriei utilizate deoarece are un număr mic de variabile simple.

III 3. b)

```
#include<fstream>
using namespace std;
ofstream fout(„bac.txt");
int main()
{
  int c1,c2,c3,nr;
  for(c1=1;c1<=9;c1=c1+2)
    for(c2=1;c2<=9;c2=c2+2)
     for(c3=1;c3<=9;c3=c3+2)
       { nr=c1*100000+c2*10000+c3*
       1000+c3*100+c2*10+c1;
         fout<<nr<<endl;
       }
  return 0;
}
```

TEZA 20

I 1. c **I 2.** b **I 3.** d **I 4.** c **I 5.** d

II 1.

```
int nr=0,i;
for(i=0;i<30;i++)
 if(strcmp(e[i].l1,"limba
engleza")==0|| strcmp(e[i].l2,
"limba engleza")==0)
nr++;
cout<<nr;
```

II 2.

```
if(i==j)
  A[i][j]=1;
else
  if(i+j==3)
    A[i][j]=7;
   else
    A[i][j]=j+1;
```

II 3. a) 5

II 3. b) Pentru orice numar prim impar se va afișa numărul care a fost citit.

II 3. c)

```
#include <iostream>
using namespace std;
int main()
{   int a,m,d;
    m=0;
    cin>>a;
    for(d=a;d>=2;d--)
        if(a%d==0)
         if(d%2==1)
           m=d;
           else
            m=2*d;
           cout<<m;
    return 0;
}
```

II 3. d)

citește a (a număr natural nenul)
m ←0,d←a
cât timp d>=2 execută
  dacă a%d=0
    dacă d%2=1
     atunci
       m ←d
     altfel
      m ←2*d
  d←d–1
 scrie m

III 1.

```
#include <iostream>
#include<cstring>
#include<cctype>
using namespace std;
int main()
{   char s[21];int prod=1,nr,i;
    cin>>s;
    i=0;
    while(i<strlen(s))
    {nr=0;
    if(isdigit(s[i])!=0)
     { while(isdigit(s[i])!=0&&i<
      strlen(s))
        {   nr=nr*10+(s[i]-'0');
            i++;
        }
        prod*=nr;}
        i++;
    }
    cout<<prod;
    return 0;
}
```

III 2.

```
int produs(int n,int x[])
{int prodmax=0,i,j;
 for(i=1;i<n;i++)
  for(j=i+1;j<=n;j++)
   if(x[i]*x[j]>prodmax)
     prodmax=x[i]*x[j];
 return prodmax;
}
```

III 3. a) Programul citește numerele din fișier pe rând, verifică pentru fiecare număr dacă prima și ultima cifră sunt egale și în caz afirmativ reține frecvența numărului interior al numărului testat în vectorul f. Se afișează numerele interioare utilizând vectorul de frecvență. Programul este eficient ca timp de execuție deoarece obține numerele interioare la o singură parcurgere a fișierului, simultan cu citirea.

III 3. b)

```
#include <iostream>
#include<fstream>
using namespace std;
ifstream fin(„bac.in");
int f[1000];
int main()
{   int x,uc,p,nr;
    while(fin>>x)
    {   p=1;nr=0;
        uc=x%10;x=x/10;
        while(x>9)
        {   nr=nr+(x%10)*p;// nr-
  număr interior al lui x
            p*=10;
            x/=10;
        }
        if(uc==x)
            f[nr]++;
   }
   p=0;
   for(x=999;x>=0;x--)
     if(f[x]!=0)
      {cout<<x<<" „;
       p=1;
      }
  if(p==0)
    cout<<"Nu exista";
  return 0;
}
```

TEZA 21

I 1. c **I 2.** c **I 3.** d **I 4.** d **I 5.** b

II 1.

```
int i;
for(i=0;i<200;i++)
if(strcmp(c[i].
   probaE_d,"Informatica")==0)
  if(c[i].nota>9)
    cout<<c[i].nume;
```

II 2.

```
strncpy(x,s,4);x[4]=NULL;
strcpy(t+7,x);
```

II 3. a) 14;

II 3. b) Pentru orice set de 5 numere pare programul afișează valoarea 10. De exemplu: un set poate fi 5 12 14 6 8 20. Al doilea set poate fi: 10 11 4 5 4 7 3 6 3 8 7 sau 4 5 6 9 27.

II 3. c)

```
#include <iostream>
using namespace std;
int main()
{   int n,a,s,d,i;
    s=0;d=2;
    cin>>n;
    for(i=1;i<=n;i++)
    {   cin>>a;
        if(a%d==0)
            s=s+d;
        else
        {   s++;
            d++;
        }
    }
   cout<<s;
   return 0;
}
```

II 3. d)

citește n (n număr natural nenul)
s ←0; d←2; i ←1
cât timp i<=n execută
 citește a
 dacă a%d=0 atunci
  s←s+d
 altfel
  s←s+1
  d←d+1
 i ←i+1
 scrie s

III 1.

```
#include <iostream>
using namespace std;
int main()
{   float s=0; int n, A[40]
[40],mini,i,j,ordonat,nr=0;
    cin>>n;
    for(i=0;i<n;i++)
      for(j=0;j<n;j++)
        cin>>A[i][j];
    for(j=0;j<n;j++)
      { ordonat=1;mini=A[0][j];
        for(i=1;i<n&& ordonat==1 ;
        i++){
            if(A[i-1][j]>A[i][j])
              ordonat=0;
              if(A[i][j]<mini)
                mini=A[i][j];
            }
         if(ordonat==1)
            {  s+=mini;
               nr++;
            }
      }
    if(nr==0)
     cout<<"Nu exista coloane
     ordonate strict crescator";
    else
     cout<<s/nr;
    return 0;
}
```

III 2.

```
void termeni(int n, int &m,int &t)
{int s;m=1,t=1;
 while(t<n)
   {s=m+t;
    m=t;
    t=s;
   }
if(t==n)
    t=m+t;
}
```

III 3. a) Programul citește numerele din fișier și verifică pentru oricare trei numere citite consecutiv x, y, z dacă y este vârf (x<y și y>z), în caz afirmativ memorează în vectorul de frecvență f numărul de apariții al vârfului y. Prin parcurgerea vectorului de frecvență se afișează în ordine crescătoare vârfurile din fișier. Programul este eficient ca timp de execuție deoarece obține vârfurile din șir și le memorează în ordine crescătoare la o singură parcurgere a fișierului, simultan cu citirea.
Programul este eficient din punct de vedere al memoriei utilizate deoarece memorează un număr minim de variabile.

III 3. b)

```
#include <iostream>
#include <fstream>
using namespace std;
ifstream fin(„bac.txt”);
int f[1000];

int main()
{ int x,y,z,i,n,exista=0;
  fin>>n>>x>>y;
  for(i=3;i<=n;i++)
   {fin>>z;
    if(x<y&&y>z)    // verificare y
    este vârf
    {  exista=1;
       f[y]++;
    }
    x=y;y=z;
   }
  if(exista==0)
   cout<<”Nu exista”;
  else
    for(i=1;i<1000;i++)
     if(f[i]>0)
      cout<<i<<” „;
  return 0;
}
```

TEZA 22

I 1. c **I 2.** c **I 3.** c **I 4.** c **I 5.** b

II 1.

```
int i,nr; float suma; char pro[20];
strcpy(pro,c[0].proba);
for(i=0;i<300;i++)
{if(strcmp(pro,c[i].proba)==0)
{suma+=c[i].punctaj;nr++;
}
else
{cout<<pro<<” ”<<suma/nr<<endl;
strcpy(pro,c[i].proba);
suma=c[i].punctaj;nr=1;
}
cout<<<<pro<<” ”<<suma/nr<<endl;
```

II 2.

```
for(i=0;i<5;i++)
  for(j=0;j<5;j++)
   if(i==j)
     A[i][j]=’a’;
   else
     if(j<i)
       A[i][j]=A[i-1][j]+1;
     else
       A[i][j]=A[i][i]+j-i;
```

II. 3 a) 3;

II 3. b) Se poate da ca exemplu orice set de cel puțin patru numere, în care nu există niciun număr care are egale cifra zecilor și cifra unităților : 78 456 6789 235 0.

II 3. c)

```
#include <iostream>
using namespace std;
int main()
{   int x,y,k;
    cin>>x;
    k=0;
    while(x!=0)
    {
        cin>>y;
        if(x/10%10==y%10 && y!=0)
            k++;
        x=y;
    }
    cout<<k;
}
```

II 3. d)

citește x (x număr natural nenul, x>9)
k ←0
repetă
  citește y (y număr natural)
    dacă x/10%10= y%10 și y≠0
    atunci
      k ←k+1
  x ←y
până când x=0
scrie k

III 1.

```
#include <iostream>
#include<fstream>
#include<cstring>
using namespace std;
int main()
{char t[501],tnou[501],c[11],ct[501],
*p,prec[501];
 char voc[]=”aeiouAEIOU”;
 cin.getline(t,501);strcpy(ct,t);
 cin>>c;
 p=strtok(t,” „);
 strcpy(prec,p);strcpy(tnou,
prec);strcat(tnou,” „);
 p=strtok(NULL,” „);
 while(p)
  { if(strcmp(p,c)==0&& strchr(voc,
  prec[0])==0 ||strcmp(p,c)!=0 )
    {  strcat(tnou,p);
       strcat(tnou,” „);
    }
    strcpy(prec,p);
    p=strtok(NULL,” „);
  }
 if(strcmp(tnou,ct)==0)
   cout<<”Textul nu s-a modificat”;
 else
   {strcpy(t,tnou);
   cout<<t;
   }
}
```

III 2.

```
int prime(int n, int a[])
{
    int nr_perechi=0,x,y,i,j,rest;
    for(i=0;i<n-1;i++)
      for(j=i+1;j<n;j++)
       {x=a[i];y=a[j];
        do
        {   rest=x%y;
            x=y;y=rest;
        }
        while(rest!=0);
        if(x==1)
           nr_perechi++;
       }
    return nr_perechi;
}
```

III 3. a) Programul citește numerele din fișier și determină pentru fiecare dintre benzi (formate din 7 numere citite consecutiv) cele mai mici două valori **min1** și **min2** (min1 < min2). La schimbarea unei benzi se afișează cele două minime din banda anterioară sau valoarea 0 (dacă nu există două minime diferite). Programul este eficient ca timp de execuție deoarece obține minimele și le afișează la o singură parcurgere a fișierului, simultan cu citirea. Programul este eficient din punct de vedere al memoriei utilizate deoarece memorează un număr minim de variabile.

III 3. b)

```
#include <iostream>
#include<fstream>
using namespace std;
ifstream fin(„bac.txt”);
int main()
{   int n,i,min1,min2,x;
    fin>>n;
    min1=min2=1000;
    for(i=1;i<=n;i++)
    { fin>>x;
      if(x<min1)
         {
          min2=min1;
          min1=x;
        }
       else
         if(x<min2&&x>min1)
           min2=x;
      if(i%7==0)
         {if(min1!=1000&&min2!=1000&&
         min1!=min2)
            cout<<min1<<” „<<min2<<” „;
          else
           cout<<0<<” „;
         min1=min2=1000;
        }
    }
return 0;
}
```

TEZA 23

I 1. c **I 2.** d **I 3.** c **I 4.** c **I 5.** c

II 1.

```
int i;
for(i=0;i<200;i++)
  if(o[i].nr_locuitori_oras/o[i].
   suprafata_oras>15)
    cout<<o[i].nume_oras<<” ”<<o[i].
    tara<<endl;
```

II 2. Se va afișa 12 itii***.

II 3 a) 12, 15, 18, 21, 24, 27, 30, 33, 36;

II 3 b) Orice set de numere a, b cu b – a = 2, k = 2, pentru care a și b au prima cifră un număr par. De exemplu, a = 20, b = 22, k = 2, se afișează 20, 21, 22.

II 3 c)

```
#include <iostream>
using namespace std;
int main()
{int x,a,b,k,t,y;
cin>>a>>b>>k;
for(x=a;x<=b;x++)
{   y=0;t=x;
    while(t!=0)
    {   y=y*10+t%10;
        t=t/10;
    }
    if(y%k==0)
        cout<<x<<” „;
}
```

II 3 d)
citește a, b, k (numere naturale nenule, a≤b, k>1)
x←a
cât timp x<=b execută
  y ←0
  t←x
    cât timp t≠0 execută
    y←y*10+t%10
    t←[t/10]
    dacă y%k=0
      atunci
       scrie x," "
  x←x+1

III 1.
```
#include <iostream>
using namespace std;
int main()
{   int n,i,j,A[50][50],nmax=0;
    cin>>n;
    for(i=0;i<n;i++)
      for(j=0;j<n;j++)
        cin>>A[i][j];
     for(i=0;i<n;i++)
     { if(A[i][i]>nmax)
        nmax=A[i][i];
        if(A[i][n-i-1]>nmax)
        nmax=A[i][n-i-1];
     }
     for(i=0;i<n;i++)
     for(j=0;j<n;j++)
        if(A[i][j]%2==0 && A[i]
        [j]<1000)
          A[i][j]=nmax;
   for(i=0;i<n;i++){
     for(j=0;j<n;j++)
       cout<<A[i][j]<<" ";
      cout<<endl;
}
return 0;
}
```

III 2.
```
void cifre(int &n, int &k)
{   int nr=0,p1=1,p2=1,cif,cn,f[10]={0};
 // vectorul f este vector de
 frecventa a cifrelor
 // nr este numarul cu cifrele impare
 ale lui n, k are cifrele pare din n
    k=0;
    cn=n;
    while(cn!=0)
    {   cif=cn%10;
        if(cif%2==0&& f[cif]==0)
        {k=k+cif*p2;
          p2*=10;
          f[cif]=1;
        }
        if(cif%2==1)
        {   nr=nr+cif*p1;
            p1*=10;
        }
        cn/=10;
 }
 if(nr==n)
     k=-1;
 else
     n=nr ;

}
```

III 3. a) Programul memorează în vectorul f frecvența literelor din mulțimea dată. Se citesc caracterele din fișier, se transformă în litere mici și se mărește frecvența lor. Se parcurge vectorul de frecvență și se afișează caracterele care au avut un număr impar de apariții în fișier. Programul este eficient ca timp de execuție deoarece la o singură parcurgere a fișierului se citesc caracterele și se obține numărul lor de apariții. Programul este eficient din punct de vedere al memoriei utilizate deoarece memorează un număr minim de variabile.

III 3. b)
```
#include <iostream>
#include<cstring>
#include<fstream>
#include<cctype>
using namespace std;
ifstream fin(„bac.in");
int f[5];
int main()
{char c,ch;int exista=0;
 while(fin>>c)
  {   ch=tolower(c);// transformare
   in litera mica
      f[ch-'a']++;
  }
 for(int i=0;i<5;i++)
    if(f[i]%2!=0)
     {   exista=1;c='a'+i;
         for(int j=1;j<=f[i];j++)
           cout<<c;
     }
 if(exista==0)
     cout<<"Nu exista";
 return 0;
 }
```


TEZA 24
I 1. b. **I 2.** c **I 3.** b **I 4.** d **I 5.** b.

II 1.
```
int i,j,suma_totala=0;
for(i=0;i<1000;i++)
 if(strcmp(f[i].gen_film,"comedie")
 ==0 || strcmp(f[i].gen_film,
 "istoric")==0)
 if(f[i].difuzare.an==2018 && f[i].
 difuzare.luna==3)
  suma_totala+=f[i].suma;
cout<<suma_totala;
```

II 2.
```
if(i==0&&j<3)
    a[i][j]=j+1;
else
   if(j==4)
    a[i][j]=-2;
else
     a[i][j]=a[i-1][j]+a[i-1][j+1];
```

II 3. a) 3

II 3. b) Pentru orice set de date de intrare de forma a,b,x,...,a în care numerele x care se citesc nu conțin măcar 2 cifre egale cu expresia a%b%10, se va afișa 0. Un astfel de set de date de intrare poate fi: 40 16 2 3 8 40.

II 3. c)
```
#include <iostream>
using namespace std;
int main()
{int a,b,n,x,t,y;
cin>>a>>b;
n=0;
do
{cin>>x;
 y=x;
 t=0;
 do
  {   if(x%10==a%b%10)
        t++;
      x/=10;
  } while(x!=0);
  if(t>=2)
    n++;
}while(y!=a);
cout<<n;
return 0;
}
```

II 3. d)
citește a, b (a, b numere naturale nenule, a>b)
n ←0
citește x (x număr natural nenul)
y←x
cât timp y≠0 execută
  t←0
  repetă
    dacă x%10=a%b%10
      atunci
        t ←t+1
    x←[x/10]
  până când x=0
  dacă t≥2
    atunci
      n←n+1
citește x
y←x
scrie n

III 1.
```
#include <iostream>
#include<cstring>
#include<cctype>
using namespace std;
int main()
{  char t[401],*p,cp[401],tt[401]="";
   int palindrom,i;
    cin.getline(t,401);
    p=strtok(t," „);
while(p)  // separararea textului
in cuvinte
{palindrom=1;strcpy(cp,p);
for(i=0;i<strlen(p)/2;i++) //
verificarea unui cuvant
    if(p[i]!=p[strlen(p)-i-1])
     { palindrom=0;
       break;
     }
    else
    { p[i]=toupper(p[i]);//
    transformare in litere mari
      p[strlen(p)-i-1]=toupper
      (p[strlen(p)-i-1]);
    }
    if(palindrom==1)
       {if(strlen(p)%2!=0)
         // se transforma
         caracterul de la
         mijlocul cuvantului
           p[strlen(p)/2]=toupper
           (p[strlen(p)/2]);
           strcat(tt,p);
       }
   else
       strcat(tt,cp);
```

```
        strcat(tt," ");
        p=strtok(NULL," ");
    }
strcpy(t,tt);
cout<<tt;
return 0;
}
```

III 2.

```
int maxim(int n, int m ,int T[30]
   [30])
{  int x,i,j,maxi=0,impar=0,par;
   for(i=0;i<n && impar==0;i++)
     if(T[i][0]%2==1)  // linia
     incepe cu numar impar
     { impar=1;
       for(j=1;j<m;j++)
         if(T[i][j]%2==0)
         { x=T[i][j];par=1;
           while(x && par) // verifica
           daca toate cifrele sunt pare
            {if(x%10%2==1)
              par=0;
              x/=10;
            }
            if(par==1 && T[i][j]>maxi)
                maxi=T[i][j];
         }
     }
   if(impar==0)
       return -1;
   else
       return maxi; }
```

III 3. a) Programul este eficient din punct de vedere al timpului de execuție deoarece la parcurgerea fișierului se citesc numerele din fișier, se determină lungimea fiecărei secvențe de numere care încep cu aceeași cifră și se compară lungimea fiecărei secvențe cu lungimea maximă. Programul este eficient din punct de vedere al memoriei utilizate deoarece utilizează un număr minim de variabile.

III 3. b)

```
#include <iostream>
#include<fstream>
using namespace std;
ifstream fin("bac.in");
int main()
{ int x,lgmax=0,nrmax_secv=0,pc,lg;
  fin>>x;pc=x/100;lg=1;
  while(fin>>x)
    if(x/100==pc)
     lg++;
    else
    {if(lg>lgmax)
     { lgmax=lg;
      nrmax_secv=1;
     }
       else
        if(lg==lgmax)
         nrmax_secv++;
     pc=x/100;lg=1;
    }
  if(lg>lgmax)
    { lgmax=lg;
      nrmax_secv=1;
    }
      else
        if(lg==lgmax)
         nrmax_secv++;
cout<<lgmax<<" "<<nrmax_secv;
return 0;
}
```

TEZA 25

I 1. b 2. d 3. c 4. c 5. c

II 1. a) algoritmul afișează numerele prime dintre primele n numere naturale care se termină cu 7. Se vor afișa 7, 17, 37, 47

b) cel mai mare număr este 9. Se vor afișa 7, 17, 37, 47, 67.

c)

```
#include<iostream>
using namespace std;
int n,i,j,x;
int main()
{ cin>>n;
X=7;
for(i=1;i<=n;i++)
{    j=2;
while(j<=n/j&&x%j!=0)
   j++;
   if(j*j>x)cout<<x<<' ';
x=x+10;
}
return 0;
}
```

d) citește n (n număr natural)

```
x←7
pentru i←1, n execută
  j←2
  dacă j≤[x/j] și x%j ≠ 0 atunci
    repetă
      j←j+1
    până când j>[x/j]sau x%j==0
  dacă j*j>x atunci
    scrie x,' '
  x←x+10
```


2.

```
if(!strchr(voc,s[i])&&!strchr(voc,
s[i+1])&&s[i]!='*'&&s[i+1]!='*'||
strchr(voc.s[i])&&strchr(voc,s[i+1]))
{strcpy(s+1,s+i+2);i--;}
```

3.

```
for(i=0;i<n;i++)
{float p=(t[i].l1+t[i].l2+t[i].l3)/3;
float aria=sqrt(p*(p-t[i].l1)*
(p-t[i].l2)*(p-t[i].l3));
if(aria>=x&&aria<=y)
cout<<t[i].l1<<' '<<t[i].l2<<'
'<<t[i].l3<<'*';
}
```

III 1.

```
#include <iostream>
using namespace std;
int n, a[11][21],i,j;
int main()
{
    cin>>n;
    for(i=1; i<=n; i++)
        for(j=1; j<2*n; j++)
            if(i==j||i+j==2*n)
                a[i][j]=1;
            else a[i][j]=2;
    for(i=1;i<=n;i++)
    {
        for(j=1;j<2*n;j++)
            cout<<a[i][j]<<' ';
        cout<<endl;
    }
    return 0;
}
```

2.

```
int kpal(int v[],int n, int k)
{
    int i,j;
    for(i=1; i<=k/2; i++)
        swap(v[i],v[k+1-i]);
    i=k+1;j=n;
    while(i<j)
    {
        swap(v[i],v[j]);
        i++;j--;
    }
    for(i=1; i<=n/2; i++)
        swap(v[i],v[n+1-i]);
    for(i=1; i<=n/2; i++)
        if(v[i]!=v[n+1-i])
            return 0;
    return 1;
}
```

3. **a)** Eficiență memorie: se utilizează doar variabile simple.

Eficiență timp: algoritm bazat pe calcularea valorilor T,L,C în O ($\sqrt{n}$) Observăm că numărul de valori din fiecare triunghi T este egal cu 1+2+3+...+(T+1)=(T+1)(T+2)/2.

Determinăm în T cel mai mic număr cu proprietatea ca $s = \sum_{i=1}^{T}(i+1)(i+2)/2 \geq n$ iar L = cel mai mic număr cu proprietatea S=s-(T+1)-T-(T-1)-...-L ≥ n și C=L, dacă S=n sau C=n-S, altfel.

b)

```
#include <iostream>
using namespace std;
int n, T, L, C, s, poz;
int main() {
    cin >> n;
    T = 0;
    while (s < n) {
        T++;
        s += (T + 1) * (T + 2) / 2;
    }
    if (s == n) cout << T << ' '
    << T + 1 << ' ' << T;
    else {
        L = T + 1;
        while (s >= n) {
            s = s - L;
            L--;
        }
        L++;
        if (s - L == n) C = L;
        else C = n - s;
        cout << T << ' ' << L << ' ' << C;
    }
    return 0;
}
```

SPECIALIZAREA ȘTIINȚELE NATURII

TEZA 1

Subiectul I

1. d **2.** b **3.** b **4.** b **5.** d

Subiectul II

4. **a)** Algoritmul calculează cifra de control a unui număr natural. Se va afișa 3.

b) Pentru orice număr nenul divizibil cu 9 se va afișa 9. Cel mai mic număr este 18, cel mai mare număr este 99.

c)
```
#include <iostream>
using namespace std;
int x,y;
int main()
{
    cin>>x;
    while(x>9)
    {
      y=0;
      while(x>0)
          {y=y+x%10;
           x=x/10;
          }
       x=y;
      }
      cout<<y;
      return 0;
 }
```

d) Se știe că cifra de control a unui număr se poate determina în funcție de restul împărțirii numărului la 9.

```
citește x (x număr natural)
┌dacă x%9=0 atunci
│ x←9
│altfel
│ x←x%9
└■
scrie x
```

5.
```
per=sqrt((x[n]-x[1])* (x[n]-x[1])+
(y[n]-y[1])* (y[n]-y[1]);
for(i=2;i<=n;i++)
per+= sqrt((x[i]-x[i-1])* (x[i]-
x[i-1])+ (y[i]-y[i-1])* (y[i]-y[i-1]);
```

6. Două tablouri pot fi: `int x[]={20, 10, 9, 7, 5, 1}, y[]={19, 10, 8, 5, 1};`

Subiectul III

1. Vom descompune n în factori primi, algoritmul va determina suma factorilor primi.

```
citeste n
s←0;d←2
cât timp(n!=1)execută
    {
        dacă(n%d==0) atunci
        {
            cat timp(n%d==0)execută
                n←n/d;
            s←s+d;
        }
        d←d+1;
    }
 scrie s;
```

2.
```
#include <iostream>
using namespace std;
int n, v[101],i,j,k;
int main()
{
    cin>>n;
    for(i=1; i<=n; i++)
        cin>>v[i];
    for(k=1; k<=n; k++)
    {
        i=1;j=n-k+1;
        while(i<=k)
        {
            cout<<v[i]<<' ,;
            i++;
        }
        while(j<=n)
        {
            cout<<v[j]<<' ,;
            j++;
        }
        cout<<endl;
    }
    return 0;
}
```

3. **a)** Vom citi succesiv numerele din fișier, vom determina valoarea cerută chiar la citire (eficiența timp). Determinăm după câți pași va fi eliminat numărul curent. Numărul maxim de pași necesari este numărul de prelucrări. Vom utiliza doar variabile simple (eficiența spațiu de memorie).

b)
```
#include <iostream>
#include<fstream>
using namespace std;
ifstream fin(„bac.txt");
int x,nrp,nrpmax,prim;
int main()
{
    while(fin>>x)
    {
        nrp=0;
        do{
      prim=1;
        if(x<=1||x%2==0&&x!=2)
prim=0;
        for(int i=3; i*i<=x; i=i+2)
        if(x%i==0)prim=0;
      if(prim==0){
            nrp++;
            x++;
        }
   }while(prim==0);
      if(nrp+1>nrpmax)nrpmax=nrp+1;
    }
    cout<<nrpmax;
    return 0;
}
```

TEZA 2

Subiectul I

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

Subiectul II

1. **a)** Algoritmul numără câte dintre valorile citite sunt puteri ale lui 2. Pentru valorile date se va afișa 3, deoarece sunt 3 puteri ale lui 2 (4, 8,16).

b) Oricare patru numere care nu sunt puteri ale lui 2. Exemplu: 12 15, 13, 45

c)
```
#include <iostream>
using namespace std;
int n,x,y;
int main()
{
    cin>>n;
    while(n>0)
    {
        cin>>x;
        while(x%2==0)
            x=x/2;
        if(x==1)
            y++;
        n--;
    }
    cout<<y;
    return 0;
}
```

d) Vom utiliza structura repetitivă „pentru – execută"

```
citește n (n număr natural)
y←0
┌pentru i←1, n execută
│ citește x
│ ┌ cât timp x%2=0 execută
│ │   x←x/2
│ └■
│ ┌ dacă x=1 atunci
│ │   y←y+1
│ └■
└■
scrie y
```

2. `XM==0&&XN==0||YM==0&&YN==0`

3. `C=(80, 78, 15, 15, 12, 9, 8, 7, 3, 3)`

Subiectul III

1. Numerele naturale care au exact trei divizori sunt pătratele perfecte de numere prime. Vom număra câte pătrate perfecte de numere prime sunt în intervalul [a,b].

```
citeste a,b
nr=0;
x←√a
y←√b
daca x*x<a atunci x←a+1
pentru i←x,y execută
      {
          prim←1;
          daca(i<=1)atunci prim←0;
          pentru j←2,i/2 executa
                daca(i%j==0)prim←0;
          daca(prim==1)atunci
   nr←nr+1;
      }
 scrie nr;
```

2. Vom parcurge vectorul și vom număra valorile egale cu valoarea curentă, fiecare valoare numărată va fi modificată.

```
#include <iostream>
using namespace std;
int n, v[51],i,j,k;
int main()
{
    cin>>n;
    for(i=1; i<=n; i++)
        cin>>v[i];
    for(i=1; i<=n; i++)
        if(v[i]!=-1)
        {
            cout<<v[i]<<' ';
            k=1;
            for(j=i+1; j<=n; j++)
                if(v[i]==v[j])
                {
                    k++;
                    v[j]=-1;
                }
            cout<<k<<endl;
            v[i]=-1;
        }
    return 0;
}
```

3. **a)** Deoarece valorile din cele două șiruri sunt numere naturale cu cel mult două cifre, vom utiliza doi vectori de frecvență, un vector pentru șirul a și un vector pentru șirul b. Apoi vom determina min{fa[i]/fb[i], cu proprietatea că fb[i]!=0, i=0,99}. Algoritmul este eficient din punct de vedere al timpului de execuție deoarece este un algoritm liniar.

b)
```
#include <iostream>
#include<fstream>
using namespace std;
ifstream fin("bac.txt");
int fa[100],fb[100],i,x,nrs,n,m;
int main()
{
    fin>>n>>m;
    for(i=1;i<=n;i++)
    {fin>>x;
    fa[x]++;
    }
    for(i=1;i<=m;i++)
    {fin>>x;
    fb[x]++;
    }
    nrs=0;
    for(i=0;i<=99;i++)
        if(fb[i]!=0)
            if(nrs==0||fa[i]/fb[i]<nrs)
            nrs=fa[i]/fb[i];
    cout<<nrs;
   return 0;
}
```

TEZA 3

Subiectul I

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

Subiectul II

1. Algoritmul construiește cel mai mare număr care se poate forma utilizând toate cifrele lui x.

a) Se va afișa 7721.

b) Două numere care conțin cifrele 2, 3, 4 de același număr de ori pot fi 3324, 2343.

c)
```
#include<iostream>
using namespace std;
int x,y,cx,i;
int main()
{cin>>x;
 y=0;
 for(i=9;i>=0;i--)
{
   cx=x;
while(cx)
{if(cx%10==i)
y=y*10+i;
cx=cx/10;
}
 }
cout<<y;
return 0;
}
```

d) Vom utiliza structura repetitivă cât „timp – execută“

citește x

(x număr întreg)

```
y←0; i←9
cât timp i≥0 execută
  cx←x
  cât timp cx>0 execută
    dacă cx%10=i atunci
      y← y*10+i
    cx←cx/10
  i←i-1
scrie y
```

2.
```
cin>>pre>>pim;
cout<<sqrt(pre*pre+pim*pim);
```

3.
```
if(i==j)cout<<2<<' ';
else cout<<1<<' ';
```

Subiectul III

1.
```
citeste n,m
nr←0;
pentru i←n,m executa
 {
     x←i;
     repeta
     {
         nr←nr+1;
         x←x/10;
     }pana cand x=0;
 }
scrie nr
```

2. Deoarece tabloul este ordonat, toate elementele egale sunt pe poziții consecutive.
```
#include <iostream>
using namespace std;
int n, v[101],w[101],i,j,k;
int main()
{
    cin>>n;
    for(i=1; i<=n; i++)
        cin>>v[i];
    w[1]=v[1];
    k=1;
    for(i=2; i<=n; i++)
        if(v[i]!=v[i-1])
        {
            k++;
            w[k]=v[i];
        }
    n=k;
    for(i=1;i<=k;i++)
            v[i]=w[i];
    for(i=1;i<=k;i++)
        cout<<v[i]<<' ';

    return 0;
}
```

3. a) Vom utiliza un vector de frecvență. Deoarece numerele sunt întregi de două cifre, vom aduna fiecare număr cu 99, astfel încât să lucrăm doar cu valori pozitive din intervalul [0,198]. La afișare avem grijă să scădem 99.

b)
```
#include <iostream>
#include<fstream>
using namespace std;
ifstream fin(„bac.txt");
int x,vf[200],i,vfmax;
int main()
{
    while(fin>>x)
    {
        vf[x+99]++;
    }
    for(i=-99;i<=99;i++)
        if(vf[i+99]>vfmax)
            vfmax=vf[i+99];
    for(i=-99;i<=99;i++)
        if(vf[i+99]==vfmax)
            cout<<i<<' ,;
    return 0;
}
```

TEZA 4

Subiectul I

1. d **2.** b **3.** d **4.** d **5.** b

Subiectul II

1. a) algoritmul determină cel mai mare divizor al lui n, diferit de n, se va afișa 13;

b) putem alege orice număr prim de o cifră 2, 3, 5, 7;

c)
```
#include<iostream>
using namespace std;
int a,b,n,c;
int main()
{ cin>>n;
a=1;b=n/2;
while(a!=0&&b>0)
{ c=n;
while(c>b)
   c=c-b;
   a=c;
b=b-1;
}
cout<<b+1;
return 0;
}
```

d) A doua structură **cat timp** determină restul împărțirii lui n la b

```
citește n  (n număr natural nenul)
a←1
b←[n / 2]
cât timp a ≠ 0 și b > 0 execută
  c←n%b
  a←c
  b←b-1
b←b+1
scrie b
```

2.
```
int i,nr=0;float media;
for(i=0;i<200;i++)
{ if(mat[i]>=5&&lrom[i]>=5 &&inf[i]>=5)
   { media=( mat[i]+lrom[i]+
   inf[i])/3.0;
     if(media>=7)
       nr++;
}
cout<<nr;
```

3.
```
{if (j==0 || j==6 ) cout<<'*'<<" ";
      else
          if ((i<=3) && (j+i==6 ||
          i==j))
                cout<<'*'<<" ";
          else  cout<<'-'<<" ";
  cout<<endl;
}
```

Subiectul III

1. citește n (n număr natural nenul)
```
găsit←0
dacă n%2≠0
   atunci
      scrie 0
   altfel
       a←n%2
       n=[n/2]
       cât timp n ≠ 0 și găsit = 0 execută
             b←n%2
             n← [n/2]
             dacă a=b
               atunci
                  scrie 0
                  găsit←1
             a←b
dacă găsit=0
  atunci
   scrie 1
```

2.
```
#include <iostream>
using namespace std;
int main()
{ int n,k,p[1000], i, a, b, r;
    for(i=1;i<n;i++)
    {
        a=i;b=n;
        while(b)
        {r=a%b;
        a=b;b=r;
        }
        if(a==1)
        {k++;
```

```
        p[k]=i;}
      }
      cout<<k<<endl;
      for(i=1;i<=k;i++)
        cout<<p[i]<<" ";
return 0;}
```

3. a) Citim k și apoi succesiv valorile din fișier, calculând pentru fiecare frecvența de apariție. Parcurgem descrescător numerele de la 99 la 0 adunând frecvențe, când suma este cel puțin k, ultimul număr pentru care am adunat frecvența este numărul căutat. Dacă în fișier sunt mai puțin de k valori se va afișa –1. Algoritmul este eficient din punct de vedere al timpului de execuție deoarece este un algoritm liniar și din punct de vedere al spațiului de memorie deoarece utilizează doar variabile simple și un vector de frecvență cu 100 de elemente.

3. b)

```
#include <iostream>
#include<fstream>
using namespace std;
ifstream fin("bac.txt");
int x,k,fr[100],i,s;
int main()
{
    fin>>k;
    while(fin>>x)
        fr[x]++;

    for(i=99; i>=0; i--)
    {
        s=s+fr[i];
        if(s>=k)
        {
            cout<<i;
            return 0;
        }
    }
    cout<<-1;
    return 0;
}
```

TEZA 5

Subiectul I

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

Subiectul II

1. a) algoritmul determină cel mai mare divizor a n numere citite succesiv, se va afișa 2.

b) putem alege oricare trei numere prime între ele 3, 5, 4.

c)

```
#include<iostream>
using namespace std;
int n,i,r,d,x;
int main()
{ cin>>n;
d=0;
for(i=1;i<=n;i++)
{cin>>x;
if(d==0)d=x;
else
do{r=x%d;x=d;d=r;}while(r);
d=x;
}
cout<<d;
return 0;
}
```

d) citește n (n număr natural)

```
d←0
pentru i←1, n execută
 citește x
 dacă d=0 atunci
    d←x
 altfel
    r←x%d
    x← d
    d←r
    cât timp r>0 execută
     r←x%d
     x← d
     d←r
    d←x
```

scrie d

2. Se determină numărul de elevi înscriși la fiecare cerc și apoi se determină valoarea maximă dintre cele 3 numere.

```
int i,nr_i=0,nr_p=0,nr_r=0,
   nr_max=0;
for(i=0;i<100;i++)
  if(cercuri[i]=='i')
      nr_i++;
  else
     if(cercuri[i]=='p')
       nr_p++;
     else
       nr_r++;
 if(nr_i>nr_p)
    nr_max=nr_i;
 else
    nr_max=nr_p;
if(nr_max<nr_r)
    nr_max=nr_r;
cout<<nr_max;
```

3.

```
 { if(i==6||i==1||j==6)
    cout<<1<<" ";
else
 if(j<=2)
  cout<<j<<" ";
  else
   if(j>=i&&j<=7-i)
    {if(i<=3 )
     cout<<i<<" ";}
   else
    if(i>3&&j<=4)
       cout<<7-i<<" ";
    else
       cout<<7-j<<" ";
cout<<endl;
    }
```

Subiectul III

1. Parcurgem cifrele lui n de la stânga la dreapta. Când găsim o cifră mai mică decât cifra dată o inserăm.

```
citeste n, c
cn←n;p←1;ok←0;
cat timp (n>9)executa
    n←n/10;
    p←p*10;

n←0;
cat timp(p>0)executa
       cif←(cn/p)%10;
    p←p/10;
       daca(c>cif&&!ok)atunci
          n←n*10+c;
        ok←1;

       n←n*10+cif;

daca(!ok)atunci
n←n*10+c;
```

2.

```
int main()
{ int n,v[1000],i,j;
cin>>n;
for(i=1; i<=n;i++)
cin>>v[i];
for(i=1; i<n;)
  if(v[i]==v[i+1])
    { for(j=i+2; j<=n; j++)
                v[j-1]=v[j];
      n--;
    }
  else i++;
cout<<n;
for(i=1; i<=n;i++)
cout<<v[i]<<" ";
  return 0;
}
```

3. a) Vom citi numerele din fișier succesiv și vom determina intervalele asociate, pentru fiecare interval asociat curent determinăm intersecția cu intervalul intersecție de până atunci. Dacă intervalul curent este [a, b], atunci determinăm maximul valorilor a și minimul valorilor b. Dacă Maxa este mai mare decât Minb, afișăm mulțimea vidă. Algoritmul este eficient din punct de vedere al timpului de execuție deoarece este un algoritm liniar și din punct de vedere al spațiului de memorie deoarece utilizează doar variabile simple.

b)

```
#include <iostream>
#include<fstream>
using namespace std;
ifstream fin(„bac.txt");
int x,y,a,b,maxa,minb,ok;
int main()
{
    fin>>x;
    a=x;
    ok=0;
    while(fin>>y)
    {
        if(y==x+1)b=y;
        else
        {
            if(ok==0)
  maxa=a,minb=b,ok=1;
            else
            {
                if(maxa<a)maxa=a;
                if(minb>b)minb=b;
            }
            a=y;
        }
        x=y;
    }
    if(ok==0)maxa=a,minb=b,ok=1;
    else
    {
        if(maxa<a)maxa=a;
        if(minb>b)minb=b;
    }
    if(minb>=maxa)cout<<maxa<<'
,<<minb;
    else cout<<"multimea vida";
    return 0;
}
```

TEZA 6

Subiectul I

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

Subiectul II

1. a) algoritmul afișează în ordine descrescătoare toate numerele din intervalul [a, b] cu proprietatea că

în reprezentarea în baza 2 numărul de cifre 1 este egal cu numărul de cifre egale cu 0. Se va afișa 12 10 9 *.

b) cel mai mare număr este 34.

c)
```
#include<iostream>
using namespace std;
int a,b,x,y,k;
int main()
{
     cin>>a>>b;
     for(x=b;x>=a;x--)
     {
          y=x;k=0;
          while(y>0)
          {
               k=k+1-2*(y%2);
               y=y/2;
          }
          if(k==0)
               cout<<x<<' ';
     }
     cout<<'*';
     return 0;
}
```

d) citește a, b (a, b numere naturale, a ≤ b)

```
y←0
pentru x←b, a, –1 execută
    y←x
    k←0
    dacă y>0 atunci
        repetă
        k←k+1–2*(y%2)
        y←[y/2]
        până când y=0
    dacă k=0 atunci
        scrie x, ' '
scrie '*'
```

2.
```
int lungime[100],latime[100],
inaltime[100],nr_cuburi=0,i;
for(i=0;i<100;i++)
    if(lungime[i]==latime[i]&&
    latime[i]==inaltime[i]
      nr_cuburi++;
cout<<nr_cuburi;
```

3.
```
{if(i%3+j%3==4) cout<<6<<" ";
     else cout<<i%3+j%3<<" ";
    cout<<endl;
   }
```

Subiectul III

1.
```
citește a,b
p←1;ca←a;n←0;
cât timp (ca>9)execută
    ca←ca/10;
      n←n+1
    p←p*10;

pentru i←0, n – 1 execută
  dacă a=b și ok=0 atunci
    Scrie i
    ok ←1
    altfel
    a←(a%10)*p+a/10

dacă ok=0 atunci
 scrie –1
```

2. Cel mai mic divizor diferit de 1 a lui n este prim, vom căuta valoarea divizorului în vectorul v prin căutare binară. Dacă îl găsim se afișează poziția divizorului în vector, în caz contrar îl vom insera în v pe poziția furnizată de căutarea binară și afișăm vectorul.

```
int main()
{   int k,n,v[30],d,ls,ld,mij,ok=0, poz;
    cin>>k;
    for(i=1;i<=k;i++)
      cin>>v[i];
        d=n;
    for(int i=2;i*i<=n;i++)
    if(n%i==0){d=i;break;}
    ls=1;ld=k;
    while(ls<=ld&&ok)
    {
        mij=(ls+ld)/2;
        if(v[mij]==d){ok=1;poz=mij;}
        else if(v[mij]>d)ld=mij-1;
        else ls=mij+1;
    }
    if(!ok)
    {
        for(int i=k;i>=ls+1;i--)
            v[i+1]=v[i];
        v[ls+1]=d;k++;
    for(i=1;i<=k;i++)
      cout<<v[i]<<" ";
    }
else
   cout<<poz;
}
```

3. a) Vom citi numerele din fișier succesiv și vom memora într-un vector de frecvență numărul de apariții a fiecărei cifre. Cifrele care apar în palindrom trebuie să apară de un număr par de ori, mai puțin cifra din mijloc. Algoritmul este eficient din punct de vedere al timpului de execuție deoarece este un algoritm liniar și din punct de vedere al spațiului de memorie deoarece utilizează doar variabile simple și un vector cu 10 elemente.

b)
```
#include <iostream>
#include<fstream>
using namespace std;
ifstream fin(„bac.txt");
int x,fr[10],exista=1,i,j,mijloc,p,n;
int main()
{
    fin>>n;
    for(i=1; i<=n; i++)
    {
        fin>>x;
        fr[x]++;
    }
    //determin cifra din mijloc daca
    exista, trebuie sa fie una singura
    mijloc=-1;
    for(i=9; i>=1; i--)
        if(fr[i]%2==1)
            if(mijloc==-1)
            {
                mijloc=i;
            }
            else
            {
                exista=0;
                break;
            }
     if(exista)
     {
          for(i=9; i>=1; i--)
          {
               for(j=1; j<=fr[i]/2;
    j++)
                    cout<<i;
          }
          if(mijloc!=-1)cout<<mijloc;
          for(i=1; i<=9; i++)
          {
               for(j=1; j<=fr[i]/2;
    j++)
                    cout<<i;
          }
     }
     if(!exista)cout<<-1;
     return 0;
}
```

TEZA 7

Subiectul I

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

Subiectul II

1a. 2664

1. b) 2 0 500 500 sau 4 3 334 103 533 133 (orice set de n numere, pentru care suma numerelor care conțin cifra y de un număr par de ori este egală cu 1000).

1. c)
```
#include <iostream>
using namespace std;
int main()
{   int e,n,y,i,x,cx,nr;
    cin>>n>>y;
    e=0;
    for(i=1;i<=n;i++)
    {
          cin>>x;
          cx=x;
          nr=0;
          while(x!=0)
          {
              if(x%10==y)
                 nr++;
                 x=x/10;
          }

          if(nr%2==0 && nr>0)
             e=e+cx;
    }
    cout<<e;
return 0;
}
```

1. d)

```
citește n, y (y număr natural, y<10, n număr natural
nenul)
 e←0
 i←1
cât timp i<=n execută
  citește x (număr natural nenul)
  cx←x
  nr←0
    cât timp x≠0 execută
      dacă x%10=y
        atunci
        nr←nr+1
    x←[x/10]
    dacă nr%2=0 și nr>0
      atunci
      e←e+cx
    i←i+1
scrie e
```

2.
```
int i,p,pmax,pmin,n;
pmin=pmax=
latura1[0]+latura2[0]+latura3[0];
for(i=1;i<n;i++)
```

```
{p=latura1[i]+latura2[i]+latura3[i];
 if(p<pmin)
   pmin=p;
 if(p>pmax)
   pmax=p;
}
cout<<pmin<<" "<<pmax;
```

3.

```
if(i==0)
 cout<<(char)('A'+j)<<" ";
 else
    if(j==0)
     cout<<(char)('A'+i)<<" ";
     else
      if(i==j)
        cout<<"A ";
      else
         if(i+j==5)
         cout<<"F ";
 else
  cout<<(char)('A'+i)<<" ";
```

Subiectul III

1.

Citeste n
i←1
k←3
┌cât timp i<=n execută
│ j ←1;
│ ┌cât timp j<=k si i<=n
│ │ ┌dacă i=n
│ │ │ atunci
│ │ │   termen=j
│ │ │
│ │ j ←j+1
│ │ i ←i+1
│ │
│ j ←k-1
│ ┌cât timp j>1 și i<=n
│ │ ┌dacă i=n
│ │ │ termen=j
│ │ │
│ │ j ←j-1
│ │ i ←i+1
│ │
│ k ←k+3
└
scrie termen;

2.

```
#include<iostream>
using namespace std;
int main()
{int prodmax=0,i,j, n, x[100];
 cin>>n;
 for(i=0;i<n;i++)
  cin>>x[i];
 for(i=0;i<n-1;i++)
  for(j=i+1;j<n;j++)
   if(x[i]*x[j]>prodmax)
     prodmax=x[i]*x[j];
 cout<< prodmax;
return 0;
}
```

3. a) Programul citește numerele din fișier pe rând, verifică pentru fiecare număr dacă prima și ultima cifră sunt egale și în caz afirmativ reține frecvența numărului interior al numărului testat în vectorul f. Se afișează numerele interioare utilizând vectorul de frecvență. Programul este eficient ca timp de execuție deoarece obține numerele interioare la o singură parcurgere a fișierului, simultan cu citirea.

3. b)

```
#include <iostream>
#include<fstream>
using namespace std;
ifstream fin(„bac.in");
int f[1000];
int main()
{   int x,uc,p,nr;
    while(fin>>x)
    {   p=1;nr=0;
        uc=x%10;x=x/10;
        while(x>9)
        {   nr=nr+(x%10)*p;// nr-
        număr interior al lui x
            p*=10;
            x/=10;
        }
        if(uc==x)
            f[nr]++;
    }
    p=0;
    for(x=999;x>=0;x--)
      if(f[x]!=0)
       {cout<<x<<" „;
        p=1;
       }
  if(p==0)
     cout<<"Nu exista";
  return 0;
}
```

TEZA 8

Subiectul I

1. c **2.** c **3.** d **4.** d **5.** c

Subiectul II

1. a) k=49 și q=35 **1. b)** a=2 și b=3

1. c)

```
#include <iostream>
using namespace std;
int main()
{   int a,b,k,q,x,y;
    k=0;q=0;
    cin>>a>>b;
    for(x=1;x<=a;x++)
        for(y=1;y<=b;y++)
         if(y%x==0)
          k+=y;
         else
          q+=y;
    cout<<k<<" „<<q;
    return 0;
}
```

1. d)

citește a,b (a≤ b, a, b numere naturale)
k←0; q←0;x←1
┌cât timp x<=a execută
│ ┌pentru y←1, b execută
│ │ ┌dacă y%x=0
│ │ │ atunci
│ │ │   k←k+y
│ │ │ altfel
│ │ │   q←q+y
│ │ │
│ │
│ x←x+1
└
scrie k," ",q

2.

```
int i;
for(i=1;i<=200;i++)
  if(loc[i]/supr[i]>15)
    cout<<i<<" ";
2. cout<<i+j+6<<" ";
```

Subiectul III

1.

Citește x
nr ←0
┌pentru i ←1, n – 1 execută
│ citește y
│ c ←y
│ ogl ←0
│ ┌cât timp c≠0 execută
│ │ ogl ←ogl*10+c%10
│ │ c ←[c/10]
│ │
│ ┌dacă x==ogl
│ │ atunci
│ │   nr ←nr+1
│ │
│ x ←y
└
scrie nr;

2.

```
#include<iostream>
using namespace std;
int main()
{int f[100],pt[100],i,c,nf=0,d,pf,
poz,ok=0,n,fp,p,v[51];
 cin>>n;
 for(i=1;i<=n;i++)
  cin>>v[i];
//nf- nr factori,f-factori primi,
p-puteri factori primi
    fp=1;
    c=v[1];
    d=2;
//determin factorii primi si puterile
lor pt v[1] si ii
// memorez in vectorul f, puterile
lor le memorez in vectorul //pt
    while(c>1)
    {pf=0;
        while(c%d==0)
        {
            c/=d;
            pf++;
        }
        if(pf!=0)
        {
            nf++;
            f[nf]=d;
            pt[nf]=pf;
        }
        d++;
    }
    poz=1;
    int j;
    for(i=2;i<=n&&fp>0;i++)
    {c=v[i];ok=0;
    // f[j] este factor prim comun
    al lui v[1] si v[i]
     for(j=poz;j<=nf;j++)
         if(c%f[j]==0)
       { pf=0;ok=1;
         while(c%f[j]==0)
         {pf++;
          c/=f[j];
         }
         if(pf<pt[j])
            pt[j]=pf;
          fp=f[j];p=pt[j];poz=j;
         break;
         }
      if(ok==0)
      {
          fp=-1;p=0;
      }
    }
cout<<fp<<" "<<p;
return 0;
}
```

3. a) Programul este eficient ca timp de execuție deoarece generează direct numerele care trebuie afișate, alegând în ordine crescătoare primele 3 cifre impare $c_1c_2c_3$, palindromul afișat va fi $c_1c_2c_3\ c_3\ c_2\ c_1$. Programul este eficient din punct de vedere al memoriei utilizate deoarece are un număr mic de variabile simple.

3. b)

```
#include<fstream>
using namespace std;
ofstream fout(„bac.txt");
int main()
{
  int c1,c2,c3,nr;
  for(c1=1;c1<=9;c1=c1+2)
    for(c2=1;c2<=9;c2=c2+2)
     for(c3=1;c3<=9;c3=c3+2)
       { nr=c1*100000+c2*10000+c3*
       1000+c3*100+c2*10+c1;
         fout<<nr<<endl;
       }
  return 0;
}
```

TEZA 9

Subiectul I

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

Subiectul II

1. a) 12, 15, 18, 21, 24, 27, 30, 33, 36;

1 b. Orice set de numere a, b cu b – a = 2, k = 2, pentru care a și b au prima cifră un număr par. De exemplu: a = 20, b = 22, k = 2, se afișează 20, 21, 22.

1. c)

```
#include <iostream>
using namespace std;
int main()
{int x,a,b,k,t,y;
cin>>a>>b>>k;
for(x=a;x<=b;x++)
{    y=0;t=x;
     while(t!=0)
     {    y=y*10+t%10;
          t=t/10;
     }
     if(y%k==0)
          cout<<x<<" „;
}
```

1. d)

citește a, b, k (numere naturale nenule, a≤b, k>1)

x←a

```
┌cât timp x<=b execută
 y ←0
 t←x
   ┌cât timp t≠0 execută
   y←y*10+t%10
   t←[t/10]
   └
    ┌dacă y%k=0
     atunci
      scrie x," "
    └
 x←x+1
└
```

2.

```
int suma[11],suma_totala=0,
suma_prec[11],i;
for(i=1;i<=10;i++)
  { suma_totala= suma_totala+suma[i];
    if(suma[i]>suma_prec[i])
      cout<<i<<" ";
  }
cout<<suma_totala;
```

3.

```
if(i%2==0)
          cout<<j<<" ";
      else
          cout<<2*i+j<<" ";
```

Subiectul III

1.

```
Citește n
m ←1
t ←1
┌ cât timp t<n execută
|    s ←m+t
|    m ←t
|    t ←s
└
 ┌dacă t=n atunci
 |   t ←m+t
 └
scrie m,t;
```

2.

```
int main()
{   int n,a[50],nr_pere-
   chi=0,x,y,i,j,rest;
    cin>>n;
    for(i=0;i<n;i++)
     cin>>a[i];
    for(i=0;i<n-1;i++)
      for(j=i+1;j<n;j++)
       {x=a[i];y=a[j];
        do  //se determina cmmdc al
   numerelor x si y
        {   rest=x%y;
            x=y;y=rest;
        }
        while(rest!=0);
        if(x==1)
           nr_perechi++;
      }
    cout<< nr_perechi;
    return 0;
}
```

3. a) Programul citește numerele din fișier și determină pentru fiecare dintre benzi (formate din 7 numere citite consecutiv) cele mai mici două valori **min1** și **min2** (min1<min2). La schimbarea unei benzi se afișează cele două minime din banda anterioară sau valoarea 0 (dacă nu există două minime diferite). Programul este eficient ca timp de execuție deoarece obține minimele și le afișează la o singură parcurgere a fișierului, simultan cu citirea. Programul este eficient din punct de vedere al memoriei utilizate deoarece memorează un număr minim de variabile.

3. b)

```
#include <iostream>
#include<fstream>
using namespace std;
ifstream fin(„bac.txt");
int main()
{    int n,i,min1,min2,x;
     fin>>n;
     min1=min2=1000;
     for(i=1;i<=n;i++)
     { fin>>x;
       if(x<min1)
          {
           min2=min1;
           min1=x;
          }
        else
          if(x<min2&&x>min1)
            min2=x;
       if(i%7==0)
          {if(min1!=1000&&min2!=1000&
          &min1!=min2)
             cout<<min1<<" „<<min2<<"
   „;
           else
            cout<<0<<" „;
          min1=min2=1000;
          }
     }
return 0;}
```

TEZA 10

Subiectul I.

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

Subiectul II.

1. **a)** Algoritmul afișează produsul factorilor primi care apar la putere impară. Se va afișa 231.

b) cel mai mic număr este 25, cel mai mare este 81

c)

```
#include<iostream>
using namespace std;
int n,p,m,x;
int main()
{
    cin>>n;
    p=1;
    m=0;
    x=2;
    while(n>=x)
    {
        if(n%x==0)
        {
            m++;
            n=n/x;
        }
        else
        {
            if(m%2!=0)
                p=p*x;
            m=0;
            x++;
        }
    }
    if(m%2!=0)p=p*x;
    cout<<p;
    return 0;
}
```

d) citește n (n număr natural, n>1)

p←1; m←0; x←2

```
┌dacă n >= x atunci
|   ┌repetă
|   |    ┌dacă n%x=0 atunci
|   |     m←m+1; n←[n/x]
|   |    altfel
|   |     ┌dacă m%2 ≠ 0 atunci
|   |      p←p*x
|   |     └
|   |    m←0; x←x+1
|   └
|  până când n<x
└
 ┌dacă m%2 ≠ 0 atunci
 |  p←p*x
 └
```

scrie p

2. `a==b&&a!=c || a==c&& a!=b|| c==b&& a!=c`

3. n=3, m=3, cele două tablouri pot fi a=(2,4,6) și b=(3,5,7).

Subiectul III.

1. citește n (n număr natural)

```
pentru i←1, n execută
  citește x
  cx←x; x←x+1
  cât timp x%2=0 execută
    x←[x/2]
  dacă x=1 atunci
    scrie cx,' '
```

2.

```
#include<iostream>
using namespace std;
int main()
{   int n,v[100],i,j,k,ok;
    cin>>n;
    for(i=1; i<=n; i++)
        cin>>v[i];
    cin>>k;
    for(i=1; i<=k/2; i++)
        swap(v[i],v[k+1-i]);
    i=k+1;j=n;
    while(i<j)
    {   swap(v[i],v[j]);
        i++;j--;
    }
    for(i=1; i<=n/2; i++)
        swap(v[i],v[n+1-i]);
    ok=1;
    for(i=1; i<=n/2; i++)
        if(v[i]!=v[n+1-i])
            ok=0;
    if(ok)cout<<"DA";
    else cout<<"NU";
    return 0;
}
```

3. a) Vom determina pentru fiecar număr printr-un algoritm eficient numărul de valori egale cu 0 de la sfârșitul lui x!, acest număr este egal cu puterea lui 5 în descompunerea în factori primi a lui x!. Determinăm maximul de zerouri și numărul de maxime printr-un algoritm clasic. Utilizăm doar variabile simple (complexitate spațiu de memorie), rezolvăm cerința în timpul citirii (complexitate timp).

b)

```
#include <iostream>
#include<fstream>
using namespace std;
ifstream fin("bac.txt");
int x,zmax, nrzmax,nrz;
int main()
{
    while(fin>>x)
    {
        nrz=0;
        while(x)
        {
            nrz+=x/5;
            x=x/5;
        }
        if(nrz>nrzmax)
        {
            nrzmax=nrz;
            zmax=1;
        }
        else if(nrz==nrzmax)zmax++;
    }
    cout<<zmax<<' ,<<nrzmax;
    return 0;
}
```


CUPRINS

BIBLIOGRAFIE

- *Programarea în limbajul C/C++ pentru liceu*, vol. 1, Emanuela Cerchez, Marinel Şerban, Editura Polirom, Iaşi
- *Programarea în limbajul C/C++ pentru liceu*, vol. 2, Emanuela Cerchez, Marinel Şerban, Editura Polirom, Iaşi
- *Programarea în limbajul C/C++ pentru liceu*, vol. 3, Emanuela Cerchez, Marinel Şerban, Editura Polirom, Iaşi
- *Fundamentele Programării, Culegere de probleme pentru clasa a IX-a*, Dana Lica, Mircea Paşoi, Editura L&S Info-Mat, Bucureşti
- *Fundamentele Programării, Culegere de probleme pentru clasa a X-a*, Dana Lica, Mircea Paşoi, Editura L&S Info-Mat, Bucureşti
- *Fundamentele Programării, Culegere de probleme pentru clasa a XI-a*, Dana Lica, Mircea Paşoi, Editura L&S Info-Mat, Bucureşti
- *Informatică Intensiv C++, Manual pentru Clasa a IX-a*, Mariana Miloşescu, Editura Didactică şi Pedagogică, Bucureşti
- *Informatică Intensiv C++, Manual pentru Clasa a X-a*, Mariana Miloşescu, Editura Didactică şi Pedagogică, Bucureşti
- *Informatică Intensiv C++, Manual pentru Clasa a XI-a*, Mariana Miloşescu, Editura Didactică şi Pedagogică, Bucureşti
- Subiecte Bacalaureat 2003, 2004, 2005, 2006
- Variante de Bacalaureat 2007, 2008, 2009
- subiecte.edu.ro- subiecte Bacalaureat
- subiecte admitere Facultatea de Informatică, Iaşi
- subiecte admitere Facultatea de Matematică-Informatică, Bucureşti
- subiecte admitere Facultatea de Matematică-Informatică, Cluj-Napoca
- pbinfo.ro
- campion.edu.ro/arhiva