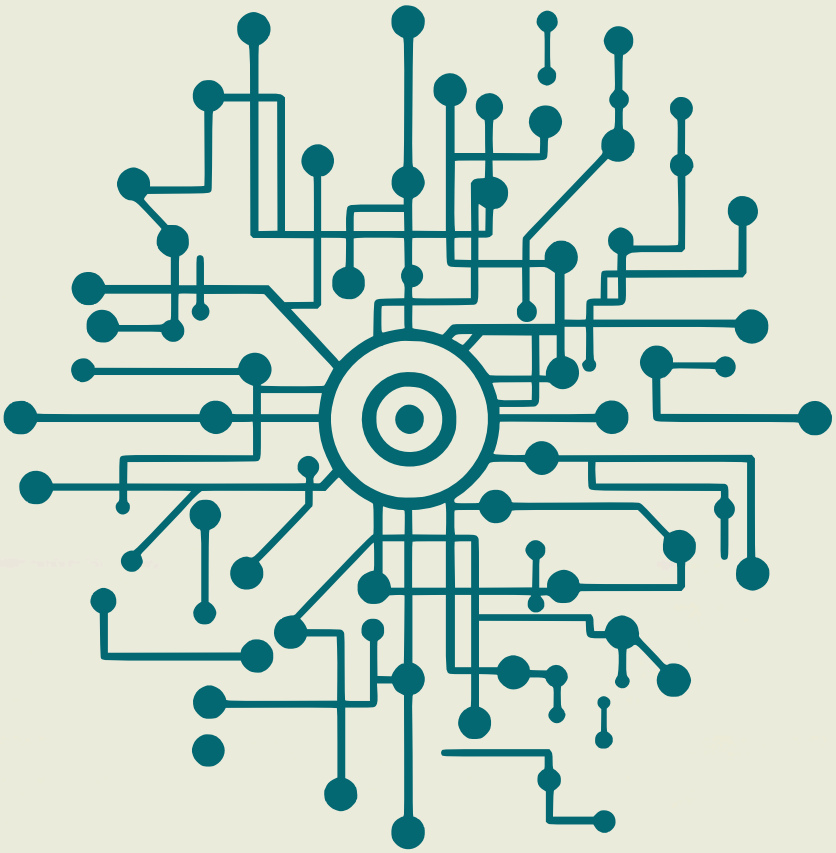
Cristina Constantinescu

ALGORITMI ESENȚIALI PENTRU BACALAUREAT

Ghid pentru rezolvarea problemelor în C++ de la Subiectul III – Problema 1• Teorie • Exemple • Subiecte rezolvate (2020–2026)



EDITURA EVOMIND

Cristina Constantinescu

Algoritmi esențiali pentru Bacalaureat

Ghid pentru rezolvarea problemelor în C++ de la Subiectul III – Problema 1• Teorie • Exemple • Subiecte rezolvate (2020–2026)

Editura EVOMIND

ISBN: 978-606-9734-64-3

2026

CUVÂNT ÎNAINTE

Această carte a apărut din dorința de a oferi elevilor de liceu un instrument clar și practic pentru pregătirea probei de Bacalaureat la informatică, în special pentru problema a de la Subiectul al III – lea.

Materialul reunește algoritmii de bază frecvent testați la examen – prelucrarea cifrelor unui număr, divizibilitate, numere prime, cel mai mare divizor comun, șirul lui Fibonacci – alături de o prezentare sistematică a subprogramelor în limbajul C++. Fiecare noțiune teoretică este însoțită de cod sursă comentat și de exemple concrete, pentru ca trecerea de la teorie la rezolvarea efectivă a unei probleme să fie cât mai naturală.

Ultima parte a cărții reunește un număr mare de subiecte rezolvate, cuprinzând variantele oficiale și testele de antrenament din perioada 2020–2026. Parcurgerea lor, în ordine sau selectiv, în funcție de nevoile fiecărui elev, oferă exercițiul necesar pentru a aborda cu încredere proba din ziua examenului.

Recomand ca această carte să fie folosită activ: rescrieți codul de mână, testați-l pe calculator, modificați-l și observați ce se schimbă. Algoritmica se învață prin exercițiu, nu doar prin citit.

Mult succes la Bacalaureat!

Cristina Constantinescu

RECOMANDAREA unui cadru universitar

Cartea "*Algoritmi esențiali pentru Bacalaureat*" reprezintă un ghid practic bine structurat, destinat elevilor care se pregătesc pentru proba de Informatică din cadrul Examenului național de bacalaureat. Autoarea urmărește să ofere o sinteză clară a algoritmilor fundamentali necesari în special pentru rezolvarea Subiectului al III-lea – Problema 1, îmbinând explicațiile teoretice cu multe subiecte rezolvate prin implementări în limbajul C++.

Unul dintre principalele puncte forte ale acestei cărți este organizarea logică a conținutului, făcând astfel învățarea mult mai naturală. Algoritmii sunt prezentați gradual, pornind de la noțiuni de bază, precum prelucrarea cifrelor unui număr natural, divizibilitate și numere prime, continuând cu algoritmul lui Euclid, șirul lui Fibonacci, determinarea valorilor minime/maxime și continuând cu detalierea utilizării subprogramelor în limbajul C++. Fiecare capitol conține explicații accesibile, exemple de cod comentate și recomandări privind modul de abordare a problemelor, ceea ce facilitează înțelegerea și aplicarea practică a conceptelor.

Deosebit de utilă este ultima secțiune, care reunește variante oficiale și teste de antrenament din perioada 2020–2026. Această colecție permite elevilor să își verifice cunoștințele pe subiecte reale și să se familiarizeze cu cerințele examenului.

Cartea promovează, de asemenea, o metodă activă de învățare, încurajând cititorii să rescrie, să testeze și să adapteze codul sursă, aspect esențial în formarea deprinderilor de programare.

În concluzie, "*Algoritmi esențiali pentru Bacalaureat*" este o resursă didactică valoroasă, caracterizată prin claritate, organizare și orientare practică. Ea se adresează în special elevilor de liceu care urmăresc obținerea unor rezultate bune la examenul național de bacalaureat, dar poate constitui și un material util pentru profesorii care pregătesc cursuri sau activități de recapitulare în domeniul algoritmicii.

Lect.univ.dr. MARIA MIROIU

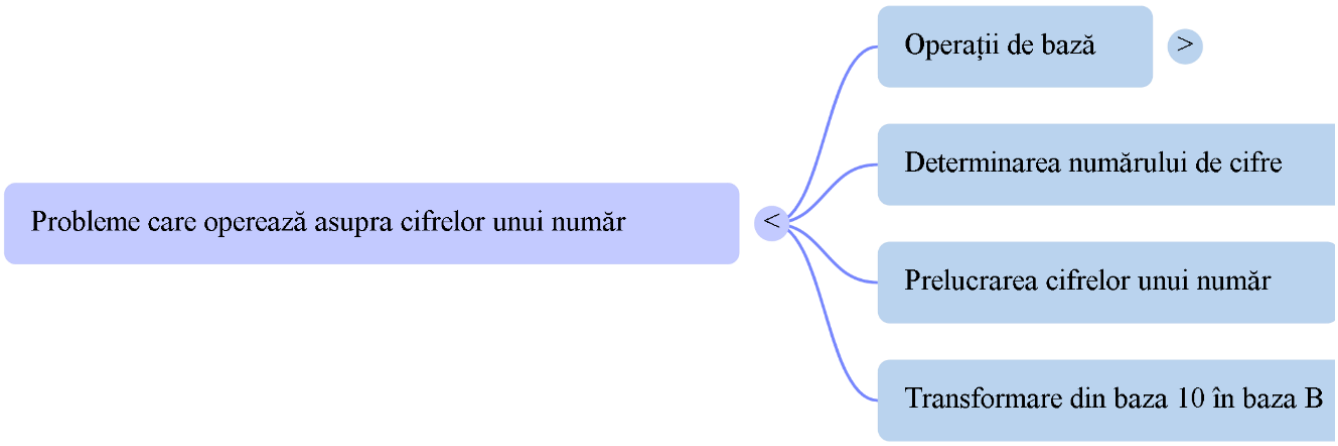
Departamentul Matematică-Informatică
Facultatea de Științe, Educație Fizică și Informatică
Universitatea Națională de Știință și Tehnologie POLITEHNICA București

CUPRINS

1. ALGORITMI ELEMENTARI – necesari pentru rezolvarea problemelor de bacalaureat de la subiectul III, problema 1.

1.1 Probleme care operează asupra cifrelor unui număr.

Pentru un număr natural n avem acces la:
n % 10 = ultima cifră a numărului;
n / 10 = numărul fără ultima cifră



➢ DETERMINAREA NUMĂRULUI DE CIFRE ALE UNUI NUMĂR.

Pentru determinarea numărului de cifre ale lui n trebuie să accesez fiecare cifră și apoi să o elimin. Algoritmul se încheie când toate cifrele au fost eliminate, adică n ajunge la valoarea 0. Se analizează, separat cazul în care n este 0 de la început.

```
cin>>n;
nr = 0;
if (n == 0) nr=1;
else while (n!=0) { nr++; n=n/10;}
cout<<nr;
```

Abordarea se face în mod similar dacă folosesc structura repetitivă cu test final.

```
cin>>n;
nr = 0;
do { nr++; n=n/10;} while (n!=0);
cout<<nr;
```

Pentru n este 12045, nr va fi 5.

➢ PRELUCRAREA CIFRELOR UNUI NUMĂR

```
cin>>n;
while (n>0)
{
        c = n%10;
        n = n/10;
   // prelucrarea cifrei c;
  //- oglinditul lui n :  o=o*10+c;
  //- determinarea unui numar x care are cifre din n, in aceeasi ordine ca in n: x=x+p*c; p=p*10; }
```

➢ TRANSFORMARE DIN BAZA 10 ÎN BAZA B

```
cin>>n>>b;
x = 0; p = 1;
while (n!=0)
{
   c = n%b;
   x = x+p*c;
   p = p*10;
   n = n/b;
}
cout<<x;
```

➢ TRANSFORMARE DIN BAZA B ÎN BAZA 10

```
cin>>n>>b;
x = 0;  p = 1;
while (n!=0)
{
   c = n%10;
   x = x+p*c;
   p = p*b;
   n = n/10;
}
cout<<x;
```

Determinării numărului de cifre, a numărului de cifre pare și impare ale unui numar nenul x.	**Numărul aparițiilor unei cifre date într-un x nenul.**

`int main()`
`{`
`  int x,nr=0,nrp=0,nri=0;`
`  cin>>x;`
`  while(x!=0)`
`  {`
`  nr=nr+1;`
`  if (x%10%2==0)`
`  nrp=nrp+1;`
`  else`
`  nri=nri+1;`
`  x=x/10;`
`  }`
`  cout<<nr<<" "<<nrp<<" "<<nri;`
`}`
	`int main()`
	`{`
	`  int x,nr=0,c;`
	`  cin>>x>>c;`
	`  while(x!=0)`
	`  {`
	`  if (x%10==c)`
	`  nr=nr+1;`
	`  x=x/10;`
	`  }`
	`  cout<<nr;`
	`}`

Construire inversului unui număr.	**Construiesc un numar din cifrele impare ale lui x .**

```cpp
int main()
{
    int x,inv=0;
    cin>>x;
    while(x!=0)
    {
        inv=inv*10+x%10;
        x=x/10;
    }
    cout<<inv;
}
```

```cpp
int main()
{
    int x,nou=0,p=1;
    cin>>x;
    while(x!=0)
    {
        if(x%10%2==1)
        {
            nou=nou+x%10*p;
            p=p*10;
        }
        x=x/10;
    }
    cout<<nou;
}
```

Determinarea cifrei minime și a numărul aparițiilor acesteia.

Inițializez cifra minima cu 10. Extrag fiecare cifră din număr și o compar cu cifra minimă.

Apar situațiile:

- Cifra este mai mică decât minima => resetez minim și numărul aparițiilor se inițializează cu 1.
- Cifra este egală cu minima => numărul aparițiilor crește cu 1.
- Elimin cifra curentă din număr
- Algoritmul se încheie când s-au eliminat toate cifrele.

```cpp
int main()
{
    int x,cmin=10,nr;
    cin>>x;
    while(x!=0)
    {
        if(x%10<cmin)
        {
            cmin=x%10;
            nr=1;
        }
        else
            if(x%10==cmin)
            nr=nr+1;
        x=x/10;
    }
    cout<<cmin<<"  "<<nr;
}
```

1.2 Divizibilitate

Spunem că numărul natural *a se divide cu d*, dacă există un număr c, astfel încât a = d*c.

Ex: 30 se divide cu numărul 5 pentru ca există un numărul 6, astfel încât 30 = 5*6.

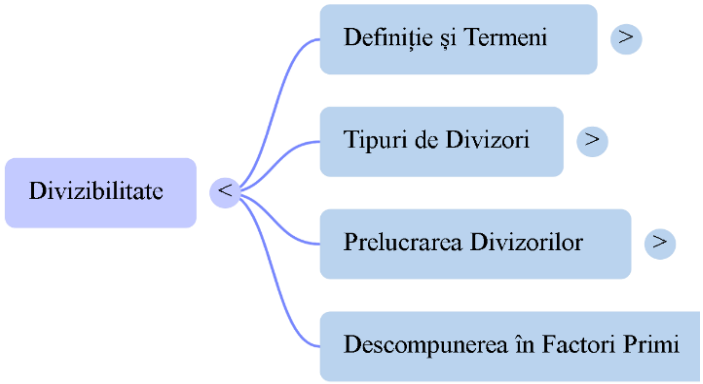
Dacă d | a, atunci d se numeşte divizor al lui a şi a se numeşte multiplu al lui d.

Divizori proprii şi improprii

Orice număr este divizibil prin 1 şi prin el însuşi. Numărul 1 şi numărul însuşi se numesc divizori improprii. Ceilalţi divizori ai numărului se numesc divizori proprii.

Ex: D6 = {1; 2; 3; 6}

1 și 6 sunt divizori improprii, iar 2 și 3 sunt divizori proprii.



➢ PRELUCRAREA DIVIZORILOR UNUI NUMĂR

a) PROPRII

```
cin>>n;
for (d = 2; d <= n/2; d++)
        if (n%d == 0)
                // prelucrarea lui d;
```

b1) TOTI DIVIZORII

```
cin>>n;
for (d = 1;d <= n; d++)
➢       if (n%d == 0)
                // prelucrarea lui d;
```

*b2)(eficient***)*

```
cin>>n;
for (d = 1; d*d<n; d++)
        if (n%d == 0)

                // prelucrarea lui d si a lui n/d;
if (d*d == n)
        // prelucrarea lui d (pentru patrate perfecte)
```

Divizorii unui număr **Afișați toți divizorii pozitivi ai unui număr natural.**	**Determinați numărul divizorilor pozitivi ai unui unui număr natural**

```
int main()
{
    int x,d;
    cin>>x;
    if (x==1)
        cout<<1;
    else
    {
        for(d=1;d<=x/2;d++)
            if(x%d==0)
            cout<<d<<' ';
        cout<<x;
    }
}
```

```
int main()
{
    int x,d,nr=1;
    cin>>x;
    for(d=2; d<=x/2; d++)
        if(x%d==0)
            nr=nr+1;
    cout<<nr;
}
```

*O abordare mai eficientă face să crească numărul divizorilor cu 2 cautându-i cât timp d*d < n, studierea cazului pentru pătrat perfect.*

➢ DESCOMPUNEREA UNUI NUMĂR ÎN FACTORI PRIMI

Afișați factorii primi și puterea la care aceștia apar în descompunerea unui număr în factori prim.

Exemplu: Pentru n=12 se va afișa:

2 La puterea 2

3 La puterea 1

```
int main()
{
    int n,d,p;
    cin>>n;
    d=1;
    while(n!=1)
    {
        d++;
        p=0;
        while(n%d==0)
        {
            p++;
            n=n/d;

        }
        if(p>0)
            cout<<d<<" La puterea "<<p<<"\n";
    }
}
```

*O a*bordarea mai eficientă va permite studierea singurului factor prim par, adică 2 și apoi divizorii impari, din 2 în 2, pornind cu 3, apoi 5 etc. În situația în care se ajunge la situația *d*d>n*, ne oprim pentru că numărul la care am ajuns este prim.

```
int d = 2, s = 0;
while (n%d == 0) { n = n/d; s++; }
if (s != 0) cout << d << ” “ << s <<’\n’;
d = 1;
while(n > 1)
{
   d += 2; s = 0;
   while (n % d == 0) { n = n/d; s++; }
   if (s != 0) cout << d << ” “ << s <<’\n’;
   if ( d*d>n && n>1) {cout << n<< “ “<<1; n = 1;}
}
```

1.3 Numere prime.

Numim număr prim orice număr natural mai mare decât 1, care are numai divizori improprii. Numere prime sunt: 2; 3; 5; 7; 11; 13; 17; 19; 23; 29; 31...
Un algoritm simplu, va testa dacă numărul are sau nu divizori.

```
cin>>n;
OK = 1; // pp ca n este numar prim
for (d = 2; d <= n/2; d++)       //SAU for (d = 2; d*d <= n && ok == 1; d++) pentru algoritm eficient
        if (n%d == 0)
                OK = 0; //n nu este prim deoarece are cel putin un divizor propriu
if (OK == 1 && n>1) // prelucrarea lui n ca numar prim
else // prelucrarea lui n ca numar neprim
```

Pentru a afla dacă un număr este prim sau nu, în mod eficient, vom verifica dacă numărul este <=1. Pentru un număr mai mare sau egal ca 2, ținem cont de faptul că singurul număr par care este prim este 2. Pentru numerele impare,testăm doar divizorii impari până la radical din număr.

```
int main()
{
    int ok,x,d;
    cin>>x;
    ok=0;
    if(x<=1)
        ok=0;
    else if(x==2)
        ok=1;
    else if(x%2==0)
        ok=0;
    else
    {
        d=3;
        ok=1;
        while(d*d<=x&&ok==1)
        {
            if(x%d==0)
                ok=0;
            d=d+2;
        }
    }
    if(ok==1)
        cout<<"Numar prim ";
    else
        cout<<"nr nu e prim";
}
```

1.4 Algoritmul lui Euclid

Algoritmul lui Euclid este o metodă eficientă de calcul al celui mai mare divizor comun(CMMDC).

CMMDC al două numere este cel mai mare număr care le divide pe ambele.

Algoritmul lui Euclid se bazează pe principiul că cel mai mare divizor comun al două numere nu se modifică dacă numărul cel mai mic este scăzut din cel mai mare.

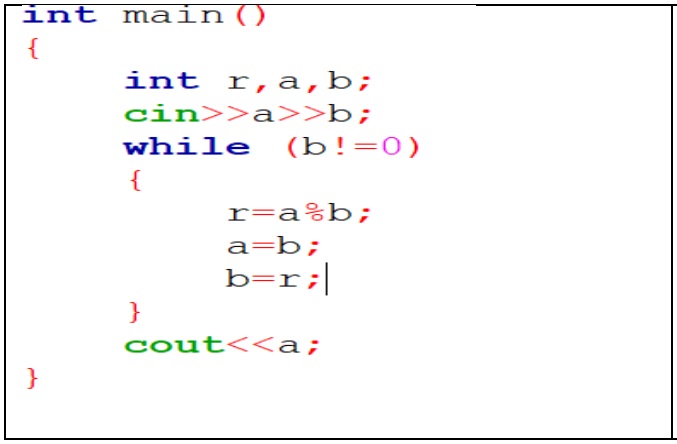
Notez (a, b) = cmmdc dintre cele două numere a și b.

$$(a, b) = \begin{cases} (a-b, b), dacă\ a > b \\ (a, b-a), dacă\ a < b \\ a, dacă\ a = b \end{cases}$$

Cum împărțirile se efectuează mai rapid, vom scrie diferențele repetate ca pe împărțiri.

$$(a, b) = \begin{cases} (b, a\ \%\ b), dacă\ b\ !=\ 0 \\ a, dacă\ b = 0 \end{cases}$$

cmmdc a două numere	cmmmc a două numere este:

	**(a * b) / cmmdc(a,b)**

Observație: Două numere care au cel mai mare divizor comun 1, se numesc *numere prime între ele* dacă cmmdc-ul lor are valoarea 1.

1.5 Șirul lui Fibonacci

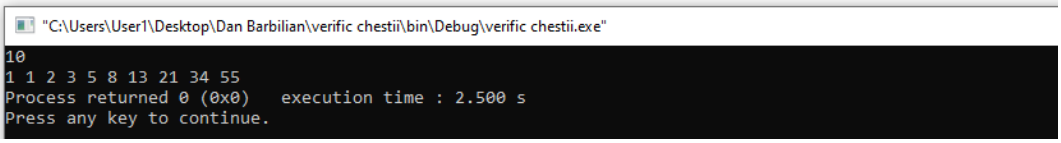
Şirul lui Fibonacci este o secvenţă de numere în care fiecare număr se obţine din suma precedentelor două din şir. Astfel, primele 10 numere ale şirului lui Fibonacci sunt:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

(primele 2 numere sunt predefinite, iar restul se obţin din suma precedentelor două: 3 = 2 + 1, 5 = 3 + 2 ...)

Pentru afișarea primilor n termeni ai șirului Fibonacci, construirea termenilor se face pas cu pas.Se pornește cu primii doi termeni. Termenul de la pasul curent se cănstruiește în funcție de termenii de la cei doi pași anteriori având grijă să păstrez termenul anterior și pe cel curent care vor permite construirea unui nou termen.

```cpp
#include <iostream>
using namespace std;
///Afișați primele n numerele ale sirului  Fibonacci

int main()
{
    int a,b,c,i,n;
    cin>>n;
    a=1;
    b=1;
    c=1;
    if(n==1)cout<<"1";
    else
    {
        cout<<"1 1 ";
        for(i=3; i<=n; i++)
        {
            c=a+b;
            a=b;
            b=c;
            cout<<c<<" ";
        }
    }
}
```



1.6 Determinare minim/maxim

Determinarea minimului/ maximului dintre anumite valori date presupune mai întâi inițializarea acestuia cu o valoare maximă/ minimă sau cu o valoare la care avem acces și apoi compararea minimului/ maximului cu valoarea curentă.

Pentru a determina maximul dintre valorile 19, 17, 58, 9, 101, 14 vom folosi o variabila *maxi* pe care o inițializez cu prima valoare adică 19, apoi îl compar pe *maxi* cu 17, apoi cu 58 care este mai mare decât *maxi*, deci devine 58. Când *maxi* se compară cu 9 rămâne neschimbat. Când îl compar cu 101, maxi devine 101. Comparându-l cu 14, rămâne neschimbat. La final variabila *maxi* reține valoarea 101.

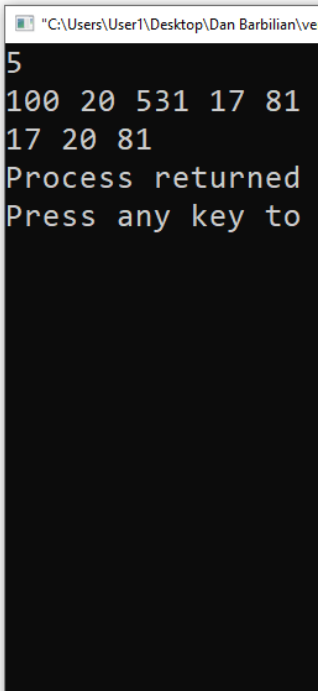
Exemple:

1.

```
#include <iostream>

using namespace std;
///Citirea a n valori și determinarea valorii maxime
int main()
{
    int n,maxi,x;
    cin>>n;
    cin>>maxi;
    for(int i=2;i<n-1;i++)
    {
        cin>>x;
        if(x>maxi)
            maxi=x;
    }
    cout<<maxi;
}
```

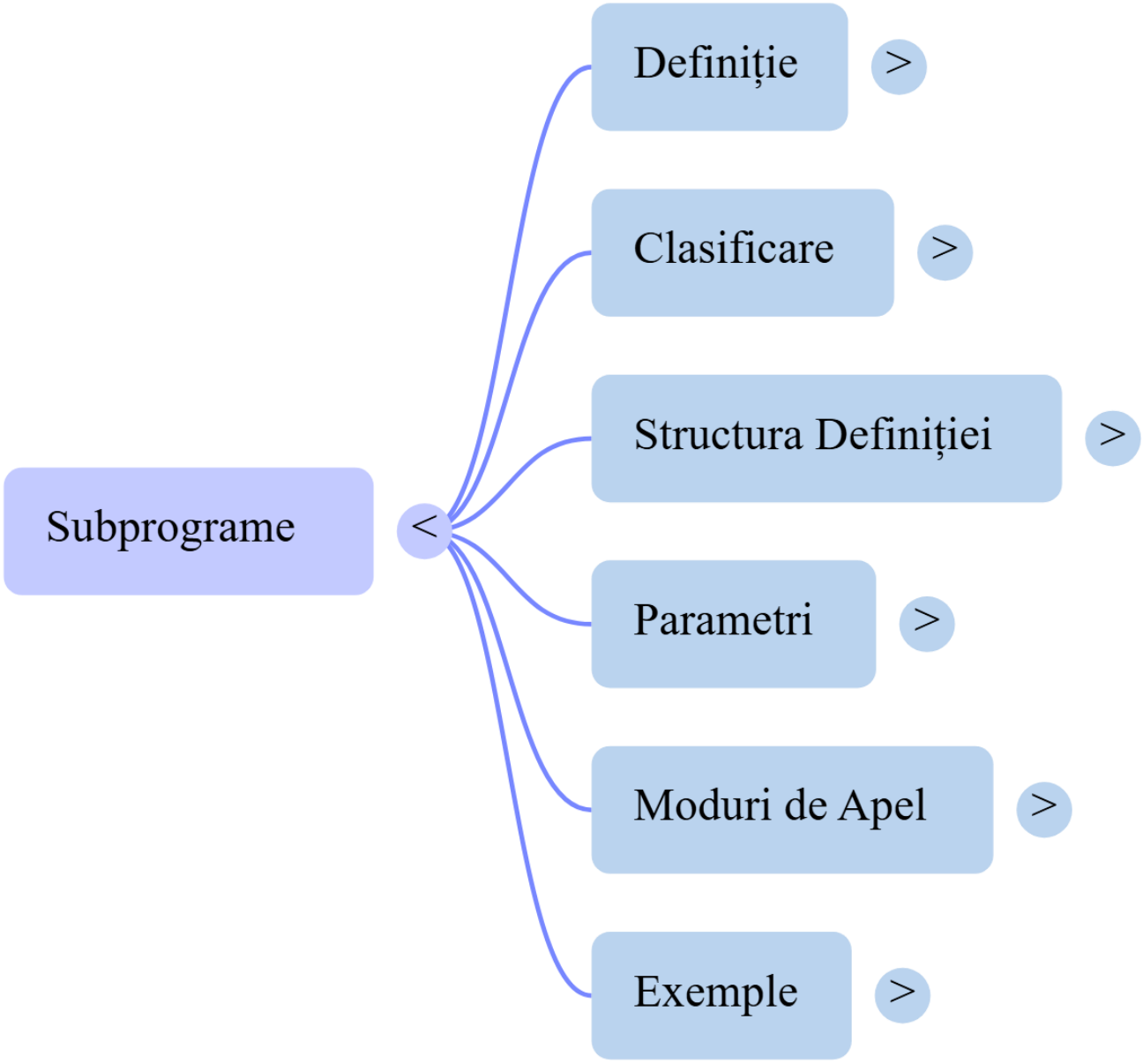
2. Pentru determinarea celor mai mici 3 valori date, știind că valorile au maxim 5 cifre, vom lua *minim1*, *minim2* și *minim 3*, inițializate la început cu 100000. Trebuie să ne asigurpm că la ficare pas avem *minim1* < *minim2* < *minim3.* Elementul curent se compară cu cel mai mic minim, adică cu *minim1*. Dacă va fi mai mic elementul curent, vom duce valoarea care era în *minim2* în *minim3*, valoarea din *minim1* o ducem în *minim2* și în *minim1* vom pune valoarea curentă. Dacă valoarea curentă nu e mai mică decât *minim1* dar este mai mică decât *minim2*, vom duce valoarea care era în minim2 în minim3 și în minim2 punem valoarea curentă, iar dacă mici această situație nu e îndeplinită, vom compara valoarea curentă cu *minim3* și dacă este mai mică valoarea curentă, o ducem în *minim3*.

```cpp
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int n;
    cin >> n;
    int minim1 = 100000, minim2 = 100000, minim3 = 100000;
    for(int i = 1; i <= n; ++i){
        int x;
        cin >> x;
        if(x < minim1){
            minim3 = minim2;
            minim2 = minim1;
            minim1 = x;
        }
        else if(x < minim2){
            minim3 = minim2;
            minim2 = x;
        }
        else if(x < minim3){
            minim3 = x;
        }
    }
    cout << minim1 << ' ' << minim2 << ' ' << minim3;
    return 0;
}
```



2. Subprograme

Subprogramele (funcțiile) sunt părți identificabile prin nume ale unui program.



În arhitectura sistemelor software complexe, subprogramele reprezintă unități logice de execuție, clar identificabile prin intermediul unui nume. Utilizarea subprogramelor permite o descriere modulară a programului. Modularizarea prin funcții permite împărțirea programului în entități mai clare, reutilizabile și mai ușor de testat.

2.1 Clasificarea subprogramelor:

- *Subprograme standard*. Utilizarea lor presupune includerea unui fisier. De exemplu, sqrt(x) întoarce radicalul numărului x. Apelul acestei funcții se poate face dacă există *#include <cmath>* sau *#include <bits/stdc++.h>* în cadrul programului.
- *Subprograme definite de utilizator* sunt scrise de programator și includ definirea și apelul lor.

Subprogramele definite de utilizator vor conține:

✓ ***Definiţia*** conţine **antetul funcţiei** şi **corpul acesteia**.

tip_returnat *nume_funcţie* (lista parametrilor formali) //antet
{
instrucţiune compusă; // corpul funcţiei
}

Tip_returnat	- Reprezintă tipul rezultatului calculat şi returnat de funcţie sau tipul **void** (nu returnează niciun rezultat). O funcţie returnează cel mult un rezultat. - Tipul rezultatului returnat poate fi orice tip de date cu excepţia vectorilor. - Dacă tipul rezultatului este **diferit de void**, corpul funcţiei trebuie să conţină cel puţin o instrucţiune **return** de tipul: *return expresie;* - Tipul expresiei calculată şi returnată de funcţie trebuie să fie de acelaşi tip cu **tip_returnat** sau poate fi un tip care se convertește implicit la tipul rezultatului.
Nume_funcţie	Un identificator ce reprezintă numele dat funcţiei.
Lista parametrilor formali	Reprezintă o listă de declaraţii de parametri **formali**, separaţi prin virgulă și precedați de tipul lor. În cadrul unui subprogram, lista parametrilor formali poate să lipsească.
Instrucţiune compusă	Poate conţine declaraţii de variabile (variabile locale) şi instrucţiuni propriu-zise.

Parametri formali sunt de trei tipuri:

- *Intrare.* Parametri formali de intrare vor prelua valori din programul unde este apelat subprogramul și le vor folosi în cadrul subprogramului.
- *Ieșire*. Sunt precedați de & și va reprezenta adresa variabilei ca parametru formal. Tablourile nu sunt precedate de &, ele reprezentând prin numele lor adresa unde li se alocă spațiu) Parametri formali de ieșire vor păstra valoarea calculată în cadrul programului și o vor transmite mai departe programului care apelează subprogramul respectiv.
- *Intrare – ieșire. V*or prelua o valoare din programul unde este apelat subprogramul, o vor folosi în cadrul subprogramului și aceasta poate fi modificată și transmisă mai departe programului care apelează subprogramul respectiv.

✓ ***Apelul*** funcţiei se realizează în cadrul programului.

După modul de apel, subprogramele se clasifică:

➢ *Subprograme apelate ca instrucțiuni procedurale* sunt apelate printr-o instrucțiune procedurală. Exemplu: citire(); calcul (x); Aceste funcții sunt definite astfel încât să permită realizarea unor instrucțiuni, eventual să transmită rezultate prin paramatri.

- ***Apelul*** funcţiei se realizează în cadrul programului:
  ***nume_funcţie* (); //**dacă funcția nu are parametri formali.
  ***nume_funcţie* (lista parametrilor actuali); //**dacă funcția nu parametri formali.

➢ *Subprograme apelate ca operanzi* apar în cadrul unei expresie.
Exemplu: cout<< sqrt(100); x = 2 * sqrt(a); Astfel de funcții sunt definite astfel încât să permită întoarcerea unui rezultat prin numele lor.

- ***Apelul*** funcţiei se realizează ca mai sus, dar în cadrul **unei expresii**.

Observații:

- *Numărul, tipul și ordinea parametrilor formali și actuali ale același subprogram trebuie să coincidă.*
- *Parametrul actual corespunzător unui parametru formal de ieșire va fi întotdeauna variabilă.*

Arhitectura unei funcții se adaptează cerințelor algoritmice. Mai jos vom găsi configurații fundamentale ale funcțiilor:

1. **Fără parametri**
2. **Fără parametri cu returnare**
3. **Cu parametru de intrare pentru folosirea acestuia în cadrul subprogramului**
4. **Parametru de intrare folosit pentru returnarea unei valori**
5. **Parametri multipli**
6. **Returnare prin parametru**
7. **Returnare prin parametru, dependent de alt parametru**
8. **Parametru de intrare-ieșire**

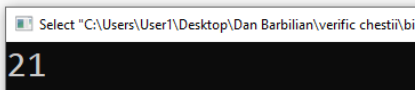
2.2 Exemple

2.2.1. Funcție fără parametri care să afișeze pe ecran un mesaj.

```
#include <bits/stdc++.h>
using namespace std;

void afisare()///antetul subprogramului
///blocul functiei
{
    cout<<"Informatica este frumoasa!";
}

int main()
{
    afisare();///apelul functiei
    return 0;
}
```

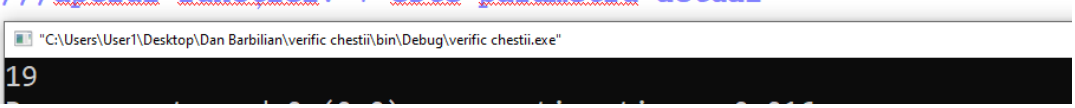
2.2.2. Funcție fără parametri care să returneze o valoare.

```
#include <bits/stdc++.h>
using namespace std;

int afisare()///antetul subprogramului
///blocul functiei
{
    return 2*3+15;///returneaza o valoare
}

int main()
{
    cout << afisare();///apelul functiei
    return 0;
}
```



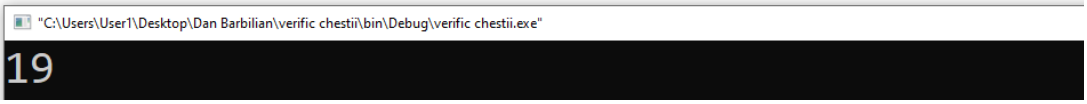
2.2.3. Funcție cu parametru care să calculeze să să afișeze o expresie în funcție de o valoare dată.

```
#include <bits/stdc++.h>
using namespace std;

void afisare(int x)///antetul subprogramului. x parametru formal de intrare
///blocul functiei
{
   cout << 2*x+5;///afiseaza expresia in functie de valoarea preluata din x
}

int main()
{
    afisare(7);///apelul functiei. 7 este parametru actual
    return 0;
}
```



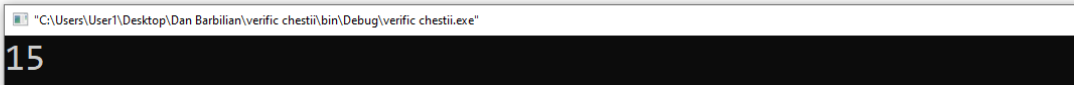
2.2.4. Funcție cu parametru care să returneze o expresie în funcție de o valoare dată.

```
#include <bits/stdc++.h>
using namespace std;

int afisare(int x)///antetul subprogramului. x parametru formal de intrare
///blocul functiei
{
   return 2*x+5;///returneaza prin numele functie expresia in functie de x
}

int main()
{
    cout << afisare(7);///apelul functiei. 7 este parametru actual
    return 0;
}
```



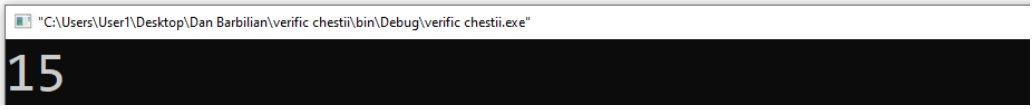
2.2.5. Funcție cu doi parametri care să returneze o expresie în funcție de 2 valori date.

```
#include <bits/stdc++.h>
using namespace std;

int afisare(int x, int y)///antetul subprogramului. x si y parametri formali de intrare
///blocul functiei
{
   return 2*x + y;///returneaza prin numele functie expresia in functie de x si y
}

int main()
{
    cout << afisare(7,1);///apelul functiei. 7 si 1 sunt parametri actuali
    return 0;
}
```



2.2.6. Functie care returnează o valoare prin parametru.

```
#include <bits/stdc++.h>
using namespace std;

void afisare(int &x)///antetul subprogramului. x formal de iesire
///blocul functiei
{
    x = 2*3 + 9;///returneaza valoarea expresiei prin x
}

int main()
{
    int a;
    afisare(a);///apelul functiei; rezultatul este retinut in variabila a
    cout << a;
    return 0;
}
```

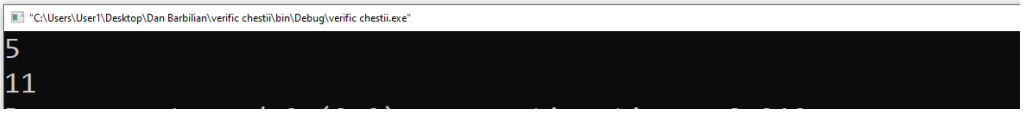


2.2.7. Functie care returnează o valoare prin parametru în funcție de altă valoare dată de un alt parametru.

```
#include <bits/stdc++.h>
using namespace std;

void afisare(int x, int &y)///antetul subprogramului. x parametru formal de intrare,
/// y parametru formal de iesire
///blocul functiei
{
    y = 2*x + 1;///returneaza valoarea expresiei prin x
}

int main()
{
    int a, b;
    cin >> a;
    afisare(a,b);///apelul functiei;
    ///rezultatul este retinut in variabila b si este calculat in functie de a
    cout << b;
    return 0;
}
```

2.2.8. Functie care returnează o valoare prin parametru în funcție de altă valoare dată preluată prin același parametru.

```
#include <bits/stdc++.h>
using namespace std;

void afisare(int &x)///antetul subprogramului. x parametru formal de intrare-iesire
///blocul functiei
{
   x = 2*x + 1;///returneaza valoarea expresiei prin x
}

int main()
{
    int a;
    cin >> a;
    afisare(a);///apelul functiei;
    ///rezultatul este retinut in variabila a si este calculat in functie de valoarea citita in a
    cout << a;
    return 0;
}
```



3. Probleme date la bacalaureat de la subiectul III, problema 1.

3.1 Subiect 2020. Test Antrenament 1

Subprogramul **putere** are trei parametri:

- **n**, prin care primește un număr natural din intervalul **[2,10⁹]**;
- **d** și **p**, prin care furnizează divizorul prim, **d**, care apare la cea mai mare putere, **p**, în descompunerea în factori primi a lui **n**; dacă există mai mulți astfel de divizori se afișează cel mai mare dintre ei.

Scrieți definiția completă a subprogramului.

Exemplu: dacă **n=10780**, atunci, în urma apelului, **d=7** şi **p=2** (**10780=$\underline{2}^2\cdot 5\cdot \underline{7}^2\cdot 11$**). **(10p.)**

```
void putere(int n, int &d, int &p)
{
    int d1=1,p1;
    p=0;
    while (n!=1)
    {
        d1++;
        p1=0;
        while(n%d1==0)
        {
            p1++;
            n=n/d1;
        }
        if(p1>=p)
        {
            p=p1;
            d=d1;
        }
    }
}
```

3.2 Subiect 2020. Test Antrenament 2

Două numere distincte **a** și **b** sunt numite **d-fii** ai unui număr natural **n** dacă **a·b=n**.

Subprogramul **fii** are un singur parametru, **n**, prin care primește un număr natural (**n∈[2,10⁹]**). Subprogramul afișează pe ecran toate perechile distincte de numere naturale cu proprietatea că sunt d-fii ai lui **n**. Fiecare pereche este afișată încadrată între paranteze rotunde, numerele din pereche fiind afișate în ordine strict descrescătoare, separate printr-un spațiu. Scrieți definiția completă a subprogramului.

Exemplu: dacă **n=12** se afișează pe ecran, nu neapărat în această ordine, **(12 1)(6 2)(4 3)** iar dacă **n=16** se afișează pe ecran **(16 1)(8 2)** **(10p.)**

```
void fii(int n)
{
    int a,b;
    for(a=n; a>sqrt(n); a--)
        if(n%a==0)
            cout<<"("<<a<<" "<<n/a<<")";

}
```

3.3 Subiect 2020. Test Antrenament 3

Subprogramul **factori** are doi parametri, **n** şi **m**, prin care primește câte un număr natural din intervalul **[1,10⁹]**. Subprogramul returnează numărul valorilor prime care se regăsesc atât în descompunerea în factori primi a lui **n**, cât și în descompunerea în factori primi a lui **m**.
Scrieți definiția completă a subprogramului.
Exemplu: dacă **n=750** şi **m=490**, atunci subprogramul returnează **2** ($750=\underline{2}\cdot 3\cdot \underline{5}^3$, $490=\underline{2}\cdot\underline{5}\cdot 7^2$). **(10p.)**

```
int factori(int n, int m)
{
    int d, p, nr=0;
    d=1;
    while (n!=1)
    {
        d++;
        p=0;
        while (n%d==0)
        {
            p++;
            n=n/d;
        }
        if (p!=0 && m%d==0)
            nr++;
    }
    return nr;
}
```

3.4 Subiect 2020. Test Antrenament 4

Două numere **a** și **b** sunt numite **generatoare** ale unui număr natural **n** dacă **a·b+[a/b]=n**, unde s-a notat cu **[c]** partea întreagă a numărului real **c**.
Subprogramul **generatoare** are un singur parametru, **n**, prin care primește un număr natural ($n\in[2,10^9]$). Subprogramul afișează pe ecran toate perechile distincte de numere naturale cu proprietatea că sunt generatoare ale lui **n** și că primul număr din pereche este par. Numerele din fiecare pereche sunt separate prin simbolul minus (**-**), iar perechile sunt separate prin câte un spațiu. Dacă nu există astfel de perechi, se afișează pe ecran mesajul **nu exista**. Scrieți definiția completă a subprogramului.
Exemplu: dacă **n=2020** se afișează pe ecran
2-1010 4-505 10-202 20-101 96-21 200-10 606-3 808-2 1010-1 **(10p.)**

```
void generatoare (int n)
{
     int i,j, ok=0;
     for(i=2; i<=n; i=i+2)
          for(j=1; j<=n; j++)
               if(i*j+i/j==n)
               {
                    cout<<i<<"-"<<j<<" ";
                    ok=1;
               }
     if(ok==0)
          cout<<"nu exista";

}
```

3.5 Subiect 2020. Test Antrenament 5

1. Un număr este scris în baza de numerație **b** (**b≤10**) dacă cifrele sale aparțin intervalului **[0,b-1]**. Subprogramul **baza** are un singur parametru, **n**, prin care primește un număr natural (**n∈[0,10⁹]**). Subprogramul returnează cea mai mică bază din intervalul **[2,10]** căreia i-ar putea corespunde scrierea lui **n**. Scrieți definiția completă a subprogramului.
Exemplu: dacă **n=50731**, subprogramul returnează numărul **8**. **(10p.)**

```
int baza(int n)
{
     int cmax=0;
     while(n!=0)
     {
          if(n%10>cmax)
               cmax=n%10;
          n=n/10;
     }
     if(cmax==0 || cmax==1)
          return 2;
     else
          return cmax+1;
}
```

3.6 Subiect 2020. Test Antrenament 6

Subprogramul **prodprim** are doi parametri:

- **n**, prin care primește un număr natural ($n \in [2, 10^9]$);
- **p**, prin care furnizează produsul divizorilor primi ai lui **n**.

Scrieți definiția completă a subprogramului.

Exemplu: dacă **n=2000**, în urma apelului **p=10**, deoarece $2000=2^4 \cdot 5^3$.

```
#include <iostream>
using namespace std;
void prodprim(int n, int &p)
{
    int d1=1,p1;
    p=1;
    while (n!=1)
    {
        d1++;
        p1=0;
        while(n%d1==0)
        {
            p1++;
            n=n/d1;
        }
        if(p1>0)
            p=p*d1;
    }
}
```

3.7 Subiect 2020. Test Antrenament 7

Subprogramul **putere** are doi parametri, **n** și **p**, prin care primește câte un număr natural ($n \in [2, 10^9]$, $p \in [0, 10^9]$). Subprogramul returnează puterea la care apare numărul **p** în descompunerea în factori primi a lui **n**, dacă **p** este număr prim, sau valoarea **-1** în caz contrar.

Scrieți definiția completă a subprogramului.

Exemplu: dacă **n=80** și **p=2**, subprogramul returnează numărul **4** ($80=2^4 \cdot 5$). **(10p.)**

```
int putere(int n, int p)
{
    int nr=0,d;
    ///verific dc p este prim
    if (p<=1)
        return -1;
    for (d=2; d*d<=p; d++)
        if (p%d==0)
            return -1;
    while(n%p==0)
    {
        nr++;
        n=n/p;
    }
    return nr;
}
```

3.8 Subiect 2020. Test Antrenament 8

Subprogramul **suma** are un singur parametru, **n**, prin care primește un număr natural ($n \in [2, 10^9]$). Subprogramul returnează suma divizorilor primi ai lui **n**. Scrieți definiția completă a subprogramului.
Exemplu: pentru **n=12** subprogramul returnează **5** (divizorii primi ai lui **12** sunt **2** și **3**). **(10p.)**

```
int suma(int n)
{
    int s=0,d=1, nr;
    while (n!=1)
    {
        d++;
        nr=0;
        while(n%d==0)
        {
            nr++;
            n=n/d;
        }
        if (nr!=0)
            s = s + d;
    }
    return s;
}
```

3.9 Subiect 2020. Test Antrenament 9

Subprogramul **suma** are doi parametri:

- **n**, prin care primește un număr natural din intervalul **[0,10⁹]**;
- **s**, prin care furnizează suma cifrelor impare distincte din scrierea acestuia.

Scrieți definiția completă a subprogramului.
Exemplu: dacă **n=4713835**, după apel **s=16** (**16=7+1+3+5**), iar dacă **n=48**, după apel **s=0**.

```
int suma(int n)
{
    int s=0,a[10]= {0},cif;
    while (n!=0)
    {
        cif = n%10;
        a[cif] = 1;
        n = n/10;
    }
    for (int i=1; i<=9; i=i+2)
        if (a[i]==1)
            s = s + i;
    return s;
}
```

3.10 Subiect 2020. Test Antrenament 10

Subprogramul **produs** are doi parametri:

- **n**, prin care primește un număr natural ($n \in [0, 10^9]$);
- **p**, prin care furnizează produsul cifrelor pare distincte din scrierea acestuia, sau **-1** dacă nu există astfel de cifre.

Scrieți definiția completă a subprogramului.

Exemplu: dacă **n=1622325**, după apel **p=12** ($12=6 \cdot 2$), iar dacă **n=122325**, după apel **p=2**. **(10p.)**

```
int suma(int n)
{
    int s=0,a[10]= {0},cif;
    while (n!=0)
    {
        cif = n%10;
        a[cif] = 1;
        n = n/10;
    }
    for (int i=0; i<=8; i=i+2)
        if (a[i]==1)
            s = s + i;
    return s;
}
```

3.11 Subiect 2020. Test 11

Subprogramul **patrate** are doi parametri, **x** și **y**, prin care primește câte un număr natural ($1 \le x \le y \le 10^9$). Subprogramul afișează pe ecran o expresie aritmetică reprezentând suma numerelor din intervalul **[x,y]** care au proprietatea că sunt pătrate perfecte, urmate de valoarea acestei sume. Termenii sumei sunt într-o ordine oarecare și sunt separați prin câte un simbol plus (**+**), iar valoarea sumei este precedată de simbolul egal (**=**), ca în exemplu. Dacă nu există niciun astfel de termen, se afișează pe ecran mesajul **nu exista**.

Scrieți definiția completă a subprogramului.

Exemplu: dacă **x=10** și **y=50** se poate afișa pe ecran **16+25+36+49=126** **(10p.)**

```
void patrate(int x,int y)
{
    int i,s=0;
    bool ok;
    ok=0;
    for(i=x; i<=y; i++)
    {
        if(sqrt(i)==int(sqrt(i)))
            if(ok==0)
            {
                ok=1;///verific daca exista cel putin un numar patrat perfect
                cout<<i;
                s=i;
            }
            else
            {
                s+=i;
                cout<<'+'<<i;
            }

        ///verific daca in intervalul[i,y]mai exista un alt patrat perfect
    }
    if(ok==0)
        cout<<"nu exista";
    else
        cout<<"="<<s;
}
```

3.12 Subiect 2020. Test Antrenament 12

Subprogramul **pDoi** are un singur parametru, **n**, prin care primește un număr natural ($n \in [1, 10^9]$). Subprogramul returnează cea mai mare valoare din intervalul **[1,n]**, cu proprietatea că este o putere a lui **2**.
Scrieți definiția completă a subprogramului.
Exemplu: dacă **n=20**, subprogramul returnează **16**. **(10p.)**

```
int pDoi(int n)
{
    int nr=0,x;
    while (1)
    {
        nr=0;
        x=n;
        while (x%2==0)
        {
            nr++;
            x=x/2;
        }
        if(x==1)
            return n;
        n--;
    }
    return nr;
}
```

3.13 Subiect 2020. Test Antrenament 13

1. Subprogramul **putere** are trei parametri:
   - **n**, prin care primește un număr natural din intervalul [2,10⁹];
   - **d** și **p**, prin care furnizează divizorul prim, **d**, care apare la cea mai mică putere, **p**, în descompunerea în factori primi a lui **n**; dacă există mai mulți astfel de divizori se afișează cel mai mic dintre ei.

   Scrieți definiția completă a subprogramului.
   Exemplu: dacă **n=10780**, atunci, în urma apelului, **d=5** și **p=1** ($10780=2^2\cdot5\cdot7^2\cdot11$). **(10p.)**

```
void putere(int n, int &d, int &p)
{
    int g = 1, p1 = 0;
    d = 1;
    p = INT_MAX;
    while (n != 1)
    {
        g++;
        p1=0;
        while (n % g == 0)
        {
            n /= g;
            p1++;
        }
        if (p1 < p && p1!=0)
        {
            p = p1;
            d = g;
        }

    }
}
```

3.14 Subiect 2020. Test Antrenament 14

Două numere **a** și **b** (**a<b**) sunt numite **divizori pereche** ai unui număr natural **n** dacă **a·b=n**.
Subprogramul **perechi** are un singur parametru, **n**, prin care primește un număr natural ($n\in[2,10^9]$). Subprogramul afișează pe ecran toate perechile distincte de numere naturale cu paritate diferită cu proprietatea că sunt divizori pereche ai lui **n**. Fiecare pereche este afișată încadrată între paranteze drepte, numerele din pereche fiind afișate în ordine strict crescătoare, separate printr-un spațiu, iar dacă nu există astfel de perechi, se afișează pe ecran mesajul **nu exista**. Scrieți definiția completă a subprogramului.
Exemplu: dacă **n=12** se afișează pe ecran, nu neapărat în această ordine, **[1 12][3 4]**
iar dacă **n=9** se afișează pe ecran **nu exista** **(10p.)**

```
void nrdiv (int n)
{
    int nr=0,ok=0;
    for (int d=1; d*d<=n; d=d+1)
        if (n%d==0 && (n/d)%2!=d%2)
        {
            cout<<"["<<d<<" "<<n/d<<"] " ;
            ok=1;
        }
    if (ok==0)
        cout<<"nu exista";
}
```

3.15 Subiect 2020. Test Antrenament 15

Subprogramul **divPrimMax** are doi parametri:

- **n**, prin care primește un număr natural ($n \in [2, 10^9]$);
- **p**, prin care furnizează cel mai mare divizor prim al lui **n**.

Scrieți definiția completă a subprogramului.

Exemplu: dacă **n=2000**, în urma apelului **p=5**, deoarece $2000=2^4 \cdot 5^3$.

```
void divPrimMax(int n, int &p)
{
    ///este alg de desc in factori primi, fara sa ne intereseze puterea
    p=1;
    while(n!=1)
    {
        p++;
        while(n%p==0)
            n=n/p;
    }
}
```

3.16 Subiect 2020. Test Antrenament 16

Subprogramul **nrDivPrimi** are un singur parametru, **n**, prin care primește un număr natural ($n \in [2, 10^9]$). Subprogramul returnează numărul divizorilor care, în descompunerea în factori primi a lui **n**, apar la o putere impară.

Scrieți definiția completă a subprogramului.

Exemplu: dacă **n=9000**, subprogramul returnează **2** ($9000=\underline{2}^3 \cdot 3^2 \cdot \underline{5}^3$). **(10p.)**

```
int nrDivPrimi(int n){
    int d=1,p,nr=0;
    while(n!=1)
    {
        d++;
        p=0;
        while(n%d==0)
        {
            p++;n=n/d;
        }
        if(p%2==1)
         nr++;
    }
    return nr;
}
```

3.17 Subiect 2020. Test Antrenament 17

Subprogramul **maxim** are un singur parametru, **n**, prin care primește un număr natural ($n \in [0, 10^9]$). Subprogramul returnează cea mai mare cifră impară din scrierea acestuia, sau **-1** dacă nu există astfel de cifre. Scrieți definiția completă a subprogramului.
Exemplu: dacă **n=5672883**, subprogramul returnează **7**. **(10p.)**

```
int maxim(int n)
{
    int cmax=-1;
    while(n!=0)
    {
        if (n%10>cmax && n%2==1)
            cmax=n%10;
        n=n/10;
    }
    return cmax;
}
```

3.18 Subiect 2020. Test Antrenament 18

Subprogramul **suma** are doi parametri:

- **n**, prin care primește un număr natural din intervalul **[0,10⁹]**;
- **s**, prin care furnizează suma cifrelor pare distincte din scrierea acestuia.

Scrieți definiția completă a subprogramului.
Exemplu: dacă **n=67638825**, după apel **s=16** (**16=6+8+2**), iar dacă **n=15**, după apel **s=0**.

```
void suma(int n, int &s)
{
    bool ap[10] = {false};///vector de aparitii
    s = 0;
    while (n > 0)
    {
        int c = n % 10;
        if (c % 2 == 0 && ap[c] == false)
        {
            s += c;
            ap[c] = true;
        }
        n /= 10;
    }
}
```

3.19 Subiect 2020. Test Antrenament 19

Subprogramul **paritate** are doi parametri:

- **n**, prin care primește un număr natural (**n∈[1,10⁹]**);
- **nr** prin care furnizează numărul de divizori naturali ai lui **n** cu aceeași paritate ca **n**.

Scrieți definiția completă a subprogramului.
Exemplu: dacă **n=20**, după apel **nr=4** (divizorii lui **20** sunt **1**, **2**, **4**, **5**, **10**, **20**).

```
void paritate(int n, int &nr)
{
     int d;
     nr=1;///n il are divizor pe el insusi
     if (n%2==0)
     {
          for (d=2; d<=n/2; d=d+2)
               if (n%d==0)
                    nr++;
     }
     else for (d=1; d<=n/2; d=d+2)
               if (n%d==0)
                    nr++;
}
```

3.20 Subiect 2020. Test Antrenament 20

1. Subprogramul **transformareBaza10** are doi parametri, **b** și **n**, prin care primește câte un număr natural (**b∈[2,10]**, **n∈[0,10⁹]**). Subprogramul returnează suma tuturor produselor de forma $c \cdot b^k$, unde **c** este cifra de pe poziția **k** în scrierea numărului **n**; pozițiile sunt numerotate de la dreapta la stânga, iar cifra unităților este pe poziția **0**.
   Scrieți definiția completă a subprogramului.
   Exemplu: dacă **b=2** și **n=10010**, subprogramul returnează **18** ($18=1 \cdot 2^4+0 \cdot 2^3+0 \cdot 2^2+1 \cdot 2^1+0 \cdot 2^0$). **(10p.)**

```
int suma(int b, int n)
{
    int s=0, p=1;
    while(n!=0)
    {
        s=s+p*(n%10);
        p=p*b;
        n=n/10;
    }
    return s;
}
```

3.21 Subiect 2020. Varianta 5

Un număr natural nenul se numește p-număr dacă are aceeași paritate cu suma divizorilor săi pozitivi.
Exemplu: 10 și 25 sunt p-numere (10 are aceeași paritate cu 18=1+2+5+10, iar 25 are aceeași paritate cu 31=1+5+25).
Subprogramul kpn, are trei parametri, a, b și k, prin care primește câte un număr natural din intervalul $[1, 10^6]$ (a≤b). Subprogramul returnează cel de al k-lea p-număr din intervalul [a,b] sau -1, dacă nu există cel puțin k astfel de numere în acest interval. Scrieți definiția completă a subprogramului.
Exemplu: dacă a=27, b=50 și k=3, atunci subprogramul returnează 34. **(10p.)**

```
int kpn(int a, int b, int k)
{
    int nr = 0;

    for (int n = a; n <= b; n++)
    {
        int s = 0;
        for (int d = 1; d * d <= n; d++)
            if (n % d == 0)
            {
                s = s + d;
                if (d != n / d)
                    s = s + n / d;
            }

        if ( n % 2 == s % 2 )
        {
            nr++;
            if (nr == k)
                return n;
        }
    }

    return -1;
}
```

3.22 Subiect 2020. Varianta 2

Subprogramul multiplu are un singur parametru, n, prin care primește un număr natural ($n \in [1, 10^4]$). Subprogramul returnează cel mai mic multiplu nenul al lui n cu proprietatea că este pătrat perfect. Scrieți definiția completă a subprogramului.
Exemplu: dacă n=72 sau n=144, subprogramul returnează numărul 144 ($144=12^2$). **(10p.)**

```
int multiplu(int n)
{
    int i=1, m=n;
    while(1){
        m = m*i;
        if (int (sqrt(m))== sqrt(m))
            return m;
        i++;
    }

}
```

3.23 Subiect 2020. Varianta 6

Subprogramul `suma` are doi parametri, `a` și `b`, prin care primește câte un număr natural din intervalul $[1, 10^4]$. Subprogramul returnează suma divizorilor naturali comuni lui `a` și `b`.
Scrieți definiția completă a subprogramului.
Exemplu: dacă `a=20` și `b=12`, atunci subprogramul returnează valoarea 7 (1+2+4=7). **(10p.)**

```
int suma(int a, int b)
{
    int d, m, s=0;
    m = min(a,b);
    for ( d=1; d <= m/2 ; d++)
        if (a%d==0 && b%d==0)
         s = s + d;
    if (a%m == 0 && b%m==0)
        s = s + m;
    return s;
}
```

3.24 Subiect 2020. Varianta Model

Subprogramul `duplicare` are doi parametri:

- `n`, prin care primește un număr natural ($n \in [1, 10^4]$);
- `d`, prin care furnizează numărul obținut prin duplicarea fiecărei cifre impare a lui `n` sau `-1` dacă acesta nu are nicio cifră impară.

Scrieți definiția completă a subprogramului.
Exemplu: dacă `n=2019`, după apel `d=201199`. **(10p.)**

```
void duplicare(int n, int &d)
{
    bool exista = false;
    d = 0;
    int p=1;
    while (n > 0)
    {
        if (n%2==1)
        {
            exista = true;
            d = d + (n%10)*p;
            p = p*10;
        }
        d = d + (n%10)*p;
        p = p*10;
        n=n/10;
    }

    if (exista == false)
        d = -1;
}
```

3.25 Subiect 2021. Varianta 4

Un număr natural **n** se numește **cub perfect** dacă există un număr natural **b**, astfel încât $n=b^3$.
Subprogramul **cuburi** are un singur parametru, **n**, prin care primește un număr natural ($n \in [1, 10^3]$).
Subprogramul afișează pe ecran, separate prin câte un spațiu, în ordine descrescătoare, primele **n** cuburi perfecte nenule.
Scrieți definiția completă a subprogramului.
Exemplu: dacă **n=5** atunci, după apel, se afișează pe ecran numerele
125 64 27 8 1 **(10p.)**

```
void cuburi(int n)
{
    for (int i = n; i >= 1; i--)
        cout << i * i * i << " ";
}
```

3.26 Subiect 2021. Varianta 1

Subprogramul **divPrim** are doi parametri:

- **n**, prin care primește un număr natural ($n \in [2, 10^9]$);
- **s**, prin care furnizează suma divizorilor primi ai lui **n** care apar la o putere impară în descompunerea în factori primi a acestuia.

Scrieți definiția completă a subprogramului.
Exemple: pentru **n=360**, după apel **s=7** ($360=2^3 \cdot 3^2 \cdot 5^1$, deci **s=2+5**), iar pentru **n=16**, după apel **s=0**.**(10p.)**

```
void divPrim(int n, int &s)
{
     int d=1,p;
     s=0;
     while(n!=1)
     {
          d++;
          p=0;
          while(n%d==0)
          {
               p++;
               n=n/d;
          }
          if(p%2==1)
               s = s + d;
     }
}
```

3.27 Subiect 2021. Varianta 7

Două numere se numesc **oglindite** dacă fiecare se obține din celălalt, prin parcurgerea cifrelor acestuia de la dreapta la stânga. Două numere se numesc **impar-oglindite** dacă numerele obținute din acestea, prin îndepărtarea tuturor cifrelor lor pare, sunt oglindite.
Subprogramul **imog** are trei parametri:

- **x** și **y**, prin care primește câte un număr natural din intervalul $[0,10^9]$;
- **rez**, prin care furnizează valoarea 1 dacă **x** și **y** sunt impar-oglindite sau valoarea 0 în caz contrar.

Scrieți definiția completă a subprogramului.
Exemplu: dacă **x=523** și **y=84356**, după apel **rez=1**,
iar dacă **x=523** și **y=84536** sau **x=523** și **y=84576** sau **x=40** și **y=86**, după apel **rez=0**. **(10p.)**

```
void imog(int x, int y, int &rez)
{
    int a = 0, b = 0, p=1;
    // Formam numarul din cifrele impare ale lui x
    while (x > 0)
    {
        if (x % 2 == 1)
            a = a * 10 + x % 10;
        x /= 10;
    }

    // Formam numarul din cifrele impare ale lui y
    //in ordine directa
    while (y > 0)
    {
        if (y % 2 == 1)
           {
             b = b  + y % 10 * p;
             p = p * 10;
           }
        y /= 10;
    }

    if (a == b)
        rez = 1;
    else
        rez = 0;
}
```

3.28 Subiect 2021. Varianta SIMULARE

Subprogramul **putere** are un parametru, **n**, prin care primește un număr natural ($n\in[2,10^9]$). Subprogramul returnează numărul prim care apare la puterea cea mai mică în descompunerea în factori primi a lui **n**. Dacă sunt mai multe astfel de numere, se returnează cel mai mic dintre acestea.
Scrieți definiția completă a subprogramului.
Exemplu: dacă **n=880**, subprogramul returnează numărul 5 ($880=2^4\cdot5\cdot11$). **(10p.)**

```
int putere(int n)
{
    int d = 1,pmin = INT_MAX, p, divi =d;
    while(n!=1)
    {
        d++;
        p = 0;
        while(n%d == 0)
        {
            p++;
            n = n/d;
        }
        if(p != 0 && p < pmin)
            {
                pmin = p;
                divi = d;
            }
    }
    return divi;
}
```

3.29 Subiect 2021. Varianta MODEL

Subprogramul **prime** are trei parametri:

- **n**, prin care primește un număr natural (**n∈[4,10^9]**);
- **x** și **y**, prin care furnizează cele mai mari două numere prime din intervalul **[1,n)**, **x<y**.

Scrieți definiția completă a subprogramului.
Exemplu: dacă **n=49**, în urma apelului **x=43**, **y=47**.

```
void prime(int n, int &x, int &y)
{
    int k = 0; bool prim;
    for (int i = n - 1; i >= 2; i--)
    {
        prim =1;
        for (int d = 2; d * d <= i; d++)
            if (i % d == 0)
            {
                prim =0;
                break;
            }
        if (prim)
        {
            if (k == 0)
            {
                y = i;
                k = 1;
            }
            else
            {
                x = i;
                break;
            }
        }
    }
}
```

3.30 Subiect 2021. Varianta Test Antrenament 1

Subprogramul **divX** are doi parametri, **n** și **x**, prin care primește câte un număr natural din intervalul **[2,50]**. Subprogramul afișează pe ecran, în ordine descrescătoare, separate prin câte un spațiu, primele **n** numere naturale nenule divizibile cu **x**.
Scrieți definiția completă a subprogramului.
Exemplu: dacă **n=4** și **x=15** în urma apelului se afișează numerele **60 45 30 15** **(10p.)**

```
void divX(int n, int x)
{
    for (int i = n; i >= 1; i--)
        cout << i * x << " ";
}
```

3.31 Subiect 2021. Varianta Test Antrenament 2

Subprogramul **factori** are doi parametri, **n** și **m**, prin care primește câte un număr natural din intervalul [1,10^9]. Subprogramul returnează numărul valorilor prime care apar la aceeași putere atât în descompunerea în factori primi a lui **n**, cât și în descompunerea în factori primi a lui **m**.
Scrieți definiția completă a subprogramului.
Exemplu: dacă **n=16500** și **m=10780**, atunci subprogramul returnează **2** (16500=2^2·3·5^3·11, 10780=2^2·5·7^2·11). **(10p.)**

```
int factori(int n, int m)
{
    int nr = 0, d=1;
    if (n>m)
        swap(m,n);
    while (n!=1)
    {
        int p1 = 0, p2 = 0;
        d++;
        while (n % d == 0)
        {
            p1++;
            n /= d;
        }

        while (m % d == 0)
        {
            p2++;
            m /= d;
        }

        if (p1 == p2 && p1 != 0)
            nr++;
        if ( d*d>n )
            if (n > 1 && m > 1 && n == m)
            {
                nr++;
                break;
            }
    }
    return nr;
}
```

3.32 Subiect 2021. Varianta Test Antrenament 3

Subprogramul **suma** are un singur parametru, **n**, prin care primește un număr natural ($n \in [1, 10^6]$). Subprogramul returnează suma divizorilor pozitivi ai lui **n** care nu sunt primi.
Scrieți definiția completă a subprogramului.
Exemplu: pentru **n=12** subprogramul returnează **23** (**23=1+4+6+12**). **(10p.)**

```
int prim(int x)
{
    if (x < 2) return 0;
    for (int d = 2; d * d <= x; d++)
        if (x % d == 0)
            return 0;
    return 1;
}

int suma(int n)
{
    int s = 0;

    for (int d = 1; d * d <= n; d++)
        if (n % d == 0)
        {
            if (!prim(d))
                s += d;

            if (d != n / d && !prim(n / d))
                s += n / d;
        }

    return s;
}
```

3.33 Subiect 2021. Varianta Test Antrenament 4

Un joc online cu **n** jetoane poate fi jucat de un grup de **k** ($k \geq 2$) jucători, numai dacă toate cele **n** jetoane pot fi distribuite în mod egal celor **k** jucători.
Subprogramul **joc** are un singur parametru, **n**, prin care primește un număr natural ($n \in [2, 10^4]$), reprezentând numărul de jetoane ale unui joc de tipul precizat. Subprogramul returnează numărul valorilor distincte pe care le poate avea **k** pentru acest joc.
Scrieți definiția completă a subprogramului.
Exemplu: dacă **n=12**, atunci subprogramul returnează numărul **5** (cele **12** jetoane se pot distribui în mod egal pentru o grupă de **2** jucători, de **3** jucători, de **4** jucători, de **6** jucători sau de **12** jucători).

```
int joc(int n)
{
    int nr = 0;

    for (int d = 1; d * d <= n; d++)
        if (n % d == 0)
        {
            if (d >= 2)
                nr++;

            if (d != n / d && n / d >= 2)
                nr++;
        }

    return nr;
}
```

3.34 Subiect 2021. Varianta Test Antrenament 5

Subprogramul **identice** are un singur parametru, **n**, prin care primește un număr natural ($n \in [10, 10^9]$). Subprogramul returnează valoarea **1**, dacă numărul **n** are toate cifrele egale, sau valoarea **0** în caz contrar. Scrieți definiția completă a subprogramului.
Exemplu: dacă **n=2222**, subprogramul returnează valoarea **1**, iar dacă **n=212**, subprogramul returnează valoarea **0**. **(10p.)**

```
int identice(int n)
{
    int c = n % 10;
    while (n > 0)
    {
        if (n % 10 != c)
            return 0;
        n /= 10;
    }
    return 1;
}
```

3.35 Subiect 2021. Varianta Test Antrenament 6

Subprogramul **numar** are trei parametri:

- **n** și **c**, prin care primește câte un număr natural ($n \in [0, 10^9]$, $c \in [0, 9]$);
- **m**, prin care furnizează numărul obținut din **n**, prin eliminarea din acesta a tuturor cifrelor egale cu **c**, sau **-1** dacă toate cifrele lui **n** sunt egale cu **c**. Cifrele nule nesemnificative sunt ignorate, ca în exemplu.

Scrieți definiția completă a subprogramului.
Exemplu: dacă **n=50752** sau **n=72** și **c=5**, după apel **m=72**, dacă **n=500** și **c=5**, după apel **m=0**, iar dacă **n=55** și **c=5**, după apel **m=-1**. **(10p.)**

```
void numar(int n, int c, int &m)
{
    int x = 0, p = 1;
    bool ok = false;
    if (n == 0)
    {
        if (c == 0)
            m = -1;
        else
            m = 0;
    }
    else
        while (n > 0)
        {
            int uc = n % 10;
            if (uc != c)
            {
                x = uc * p + x;
                p *= 10;
                ok = true;
            }
            n /= 10;
        }

    if (ok)
        m = x;
    else
        m = -1;
}
```

3.36 Subiect 2021. Varianta Test Antrenament 7

Subprogramul **afisare** are trei parametri:

- **x** și **y**, prin care primește câte un număr natural din intervalul **[0,10⁶]** (**x≤y**);
- **k**, prin care primește un număr natural (**k∈[2,10²]**).

Subprogramul afișează pe ecran, în ordine strict crescătoare, numerele din intervalul **[x,y]**, în secvențe de câte **k**, cu excepția ultimei secvențe care poate conține mai puțin de **k** numere. Fiecare secvență se încheie cu câte un simbol *, iar numerele și simbolurile sunt separate prin câte un spațiu, ca în exemplu.
Scrieți definiția completă a subprogramului.
Exemplu: dacă **x=11**, **y=21** și **k=4** se afișează pe ecran numerele de mai jos, în acest format.
11 12 13 14 * 15 16 17 18 * 19 20 21 * **(10p.)**

```
void afisare(int x, int y, int k)
{
    int cnt = 0;
    for (int i = x; i <= y; i++)
    {
        cout << i << " ";
        cnt++;
        if (cnt == k)
        {
            cout << "* ";
            cnt = 0;
        }
    }
    if (cnt != 0)
        cout << "*";
}
```

3.37 Subiect 2021. Varianta Test Antrenament 8

Subprogramul `nrfp` are doi parametri:

- `n`, prin care primește un număr natural ($n \in [2, 10^5]$);
- `m`, prin care furnizează numărul din intervalul închis `[2,n]` care are cei mai mulți factori primi; dacă există mai multe numere cu această proprietate, subprogramul îl returnează pe cel mai mare dintre ele.

Scrieți definiția completă a subprogramului.
Exemplu: dacă `n=100` atunci, în urma apelului, `m=90`. **(10p.)**

```
void nrfp(int n, int &m)
{
    int maxim = 0;

    for (int i = 2; i <= n; i++)
    {
        int x = i, nr = 0;

        for (int d = 2; d * d <= x; d++)
        {
            if (x % d == 0)
            {
                nr++;
                while (x % d == 0)
                    x /= d;
            }
        }

        if (x > 1)
            nr++;

        if (nr >= maxim)
        {
            maxim = nr;
            m = i;
        }
    }
}
```

3.38 Subiect 2021. Varianta Test Antrenament 9

1. Subprogramul **divizor** are patru parametri:

- **a**, **b** și **k**, prin care primește câte un număr natural (**a∈[0,10⁹]**, **b∈[a,10⁹]**, **k∈[1,9]**);
- **nr**, prin care furnizează numărul de valori naturale din intervalul **[a,b]** care sunt divizibile cu **k** și au ultima cifră egală cu **k**. Scrieți definiția completă a subprogramului.

Exemplu: dacă **a=3**, **b=50** și **k=4**, în urma apelului, **nr=3** (pentru numerele **4**, **24**, **44**). **(10p.)**

```
void divizor(int a, int b, int k, int &nr)
{
    nr = 0;
    int x = a;
    while (x % 10 != k)
        x++;
    while (x <= b)
    {
        if (x % k == 0)
            nr++;
        x += 10;
    }
}
```

3.39 Subiect 2021. Varianta Test Antrenament 10

Numerele naturale **x** și **y** sunt numite **în armonie** dacă suma lor aparține intervalului deschis definit de suma divizorilor lui **x**, respectiv suma divizorilor lui **y**.
Subprogramul **armonie** are doi parametri, **x** şi **y**, prin care primește câte un număr natural din intervalul $[1,10^6]$. Subprogramul returnează valoarea 1, dacă **x** și **y** sunt în armonie, sau valoarea 0 în caz contrar.
Scrieți definiția completă a subprogramului.
Exemplu: dacă **x=8**, iar **y=12** subprogramul returnează 1 ($1+2+4+8=15$, $1+2+4+6+12=25$, iar $8+12=20 \in (15,25)$), iar dacă **x=8** și **y=13**, subprogramul returnează 0 ($1+2+4+8=15$, $1+13=14$, iar $8+13=21 \notin (14,15)$). **(10p.)**

```
int armonie(int x, int y)
{
    int s_x = 0, s_y = 0;
    for (int d = 1; d * d <= x; d++)
        if (x % d == 0)
        {
            s_x += d;
            if (d != x / d)
                s_x += x / d;
        }
    for (int d = 1; d * d <= y; d++)
        if (y % d == 0)
        {
            s_y += d;
            if (d != y / d)
                s_y += y / d;
        }
    int s = x + y;
    if (s_x < s_y)
        return (s > s_x && s < s_y);
    else
        return (s > s_y && s < s_x);
}
```

3.40 Subiect 2021. Varianta Test Antrenament 11

Două numere se numesc **oglindite** dacă fiecare se obține din celălalt, prin parcurgerea cifrelor acestuia de la dreapta la stânga. Două numere se numesc **impar-oglindite** dacă numerele obținute din acestea, prin îndepărtarea tuturor cifrelor lor pare, sunt oglindite.
Subprogramul **imog** are trei parametri:
- **x** și **y**, prin care primește câte un număr natural din intervalul $[0,10^9]$;
- **rez**, prin care furnizează valoarea 1 dacă **x** și **y** sunt impar-oglindite sau valoarea 0 în caz contrar.

Scrieți definiția completă a subprogramului.
Exemplu: dacă **x=523** și **y=84356**, după apel **rez=1**,
iar dacă **x=523** și **y=84536** sau **x=523** și **y=84576** sau **x=40** și **y=86**, după apel **rez=0**. **(10p.)**

```
void imog(int x, int y, int &rez)
{
    int a = 0, b = 0, p = 1;
    if (x == 0)
        a = 0;
    else
    {
        while (x > 0)
        {
            int c = x % 10;
            if (c % 2 == 1)
            {
                a = c * p + a;
                p *= 10;
            }
            x /= 10;
        }
    }
    p = 1;
    if (y == 0)
        b = 0;
    else
    {
        while (y > 0)
        {
            int c = y % 10;
            if (c % 2 == 1)
                b = b * 10 + c;
            y /= 10;
        }
    }
    if (a == b)
        rez = 1;
    else
        rez = 0;
}
```

3.41 Subiect 2021. Varianta Test Antrenament 12

Un număr **y** este numit **frate mai mare** al unui număr **x** dacă **x** și **y** au același număr de cifre și fiecare cifră a lui **y** se poate obține din cifra aflată pe aceeași poziție în **x** adunând la aceasta valoarea **1**.
Subprogramul **frate** are doi parametri:

- **x**, prin care primește un număr natural ($x \in [0, 10^9]$);
- **y**, prin care furnizează fratele mai mare al lui **x**, sau **-1**, dacă nu se poate obține un astfel de număr.

Scrieți definiția completă a subprogramului.
Exemplu: dacă **x=1027**, după apel **y=2138**, iar dacă **x=9027**, după apel **y=-1**. **(10p.)**

```
void frate(int x, int &y)
{
    int p = 1;
    y = 0;

    if (x == 0)
    {
        y = 1;
        return;
    }

    while (x > 0)
    {
        int c = x % 10;
        if (c == 9)
        {
            y = -1;
            break;
        }
        else
        {
            y = (c + 1) * p + y;
            p *= 10;
            x /= 10;
        }
    }
}
```

3.42 Subiect 2022. Varianta 5

Subprogramul **schimb** are trei parametri:

- **n** şi **x**, prin care primeşte câte un număr natural (**n**∈ **[0,10⁸]**, **x**∈ **[1,9]**);
- **p**, prin care primește un număr natural reprezentând poziția unei cifre a numărului **n** (**0≤p**). Pozițiile cifrelor sunt numerotate de la dreapta la stânga, astfel: cifra unităților este pe poziția **0**, cifra zecilor este pe poziția **1** ș.a.m.d.

Subprogramul transformă numărul **n**, înlocuind cifra de pe poziția **p** cu cifra **x**, și furnizează numărul obținut tot prin parametrul **n**. Scrieţi definiţia completă a subprogramului.

Exemplu: dacă **n=12587**, **x=6** și **p=3**, după apel, **n=16587**. **(10p.)**

```
void schimb(int &n, int x, int p)
{
    int p10 = 1;
    for (int i = 0; i < p; i++)
        p10 *= 10;
    int cifra = (n / p10) % 10;
    n = n - cifra * p10 + x * p10;
}
```

3.43 Subiect 2022. Varianta 1

Subprogramul **secventa** are un singur parametru, **n**, prin care primeşte un număr natural (**n**∈ **[10,10⁹)**) în care nu există secvențe de mai mult de două cifre identice aflate pe poziții consecutive. Subprogramul înlocuiește în **n** fiecare secvență **22** cu câte o secvență **20** și furnizează, prin același parametru, numărul obținut. Dacă nu se înlocuiește nicio secvență, subprogramul furnizează numărul nemodificat.
Scrieţi definiţia completă a subprogramului.

Exemplu: dacă **n=202233228**, după apel **n=202033208**. **(10p.)**

```
void secventa(int &n)
{
    int m = 0, p = 1;
    while (n > 0)
    {
        int c1 = n % 10;
        n /= 10;
        if (c1 == 2 && n % 10 == 2)
        {
            m += 20 * p;
            p *= 100;
            n /= 10;
        }
        else
        {
            m += c1 * p;
            p *= 10;
        }
    }

    n = m;
}
```

3.44 Subiect 2022. Varianta 4

Subprogramul **patrate** are trei parametri:
- **n**, prin care primeşte un număr natural ($n \in [2, 10^9]$);
- **x** și **y**, prin care furnizează câte un număr natural cu proprietatea că $x^2 \cdot y^2 = n$ și $2 \leq x < y$ sau valoarea 0, prin fiecare dintre aceștia, dacă nu există două astfel de numere. Dacă sunt mai multe astfel de valori, se furnizează cele corespunzătoare unei valori minime a lui **x**.

Scrieţi definiţia completă a subprogramului.
Exemplu: pentru **n=400**, după apel, **x=2** şi **y=10**, iar pentru **n=16** sau **n=24**, după apel, **x=0** şi **y=0**. **(10p.)**

```
void patrate(int n, int &x, int &y)
{
    int r = sqrt(n);
    x = y = 0;
    if (r * r != n)
        x = y = 0;
    else
    {
        for (int i = 2; i * i < r; i++)
            if (r % i == 0)
            {
                x = i;
                y = r / i;
                break;
            }
    }
}
```

3.45 Subiect 2022. Varianta SIMULARE

Subprogramul **rest** are patru parametri:
- **x**, **y** și **n**, prin care primește câte un număr natural din intervalul **[1,10⁶]**, **x<y<n**;
- **k**, prin care furnizează cea mai mare valoare naturală din intervalul **[1,n]** pentru care atât restul împărțirii la **x**, cât și restul împărțirii la **y**, sunt egale cu **2**, sau **0** dacă nu există o astfel de valoare.

Scrieți definiția completă a subprogramului.
Exemplu: pentru **x=10**, **y=101** și **n=3000**, subprogramul returnează numărul **2022** (pentru numerele **2**, **1012** și **2022** atât restul împărțirii la **10**, cât și restul împărțirii la **101**, este **2**). **(10p.)**

```
void rest(int x, int y, int n, int &k)
{
        k = 0;
        for (int i = n; i >=1 ; i--)
            if (i%x == 2 && i%y == 2)
            {
                k = i;
                break;
            }
}
```

3.46 Subiect 2023. Varianta 7

Subprogramul **DNPI** are un singur parametru, **n**, prin care primește un număr natural ($n \in [1, 10^9]$), și afișează pe ecran, separați prin câte un spațiu, toți divizorii pozitivi impari ai lui **n** care **NU** sunt primi. Scrieți definiția completă a subprogramului.
Exemplu: dacă **n=90**, se afișează pe ecran, nu neapărat în această ordine, numerele
1 9 15 45 **(10p.)**

```
void DNPI(int  n)
{
    int d;
    for (d = 1; d <= n/2; d++)
        if (n % d == 0)
        {
            if (d % 2 == 1)
            {
                int nrDiv = 0;
                for (int i = 1; i <= d; i++)
                    if (d % i == 0) nrDiv++;

                if (nrDiv != 2) cout << d << " ";
            }
        }
    if (n%2 == 1)
        cout<<n;
}
```

3.47 Subiect 2023. Varianta 5

Un număr natural nenul, **n**, se numește număr **abundent** dacă **S(n)/n>S(k)/k**, pentru orice număr natural nenul **k** (**k≤n-1**), unde s-a notat cu **S(i)** suma divizorilor pozitivi ai numărului natural nenul **i**.
Subprogramul **abundent** are un singur parametru, **n**, prin care primește un număr natural ($n \in [2, 10^6]$). Subprogramul returnează valoarea **1**, dacă **n** este un număr abundent, sau valoarea **0**, în caz contrar. Scrieți definiția completă a subprogramului.
Exemplu: pentru **n=6**, subprogramul returnează valoarea **1** (**S(6)/6=2**, iar cel mai mare raport obținut pentru valori strict mai mici decât **6** este **S(4)/4=1.75**), iar pentru **n=7** sau **n=8**, subprogramul returnează valoarea **0** (**S(7)/7=1.14**, **S(8)/8=1.87**). **(10p.)**

```
int abundent(int n)
{
    int Sn = 0;
    // S(n)
    for (int d = 1; d * d <= n; d++)
        if (n % d == 0)
        {
            Sn += d;
            if (d != n / d)
                Sn += n / d;
        }
    // verific toate valorile k < n
    for (int k = 2; k < n; k++)
    {
        int Sk = 0;
        //  S(k)
        for (int d = 1; d * d <= k; d++)
            if (k % d == 0)
            {
                Sk += d;
                if (d != k / d)
                    Sk += k / d;
            }
        // compar S(n)/n si S(k)/k
        if (Sn * k <= Sk * n)
            return 0;
    }
    return 1;
}
```

3.48 Subiect 2023. Varianta 6

Subprogramul **Putere** are trei parametri:

- **n**, prin care primește un număr natural ($n \in [2, 10^9]$);
- **x** și **p**, prin care furnizează două numere naturale cu proprietatea că $n=x^p$, iar **x** este cel mai mic număr cu această proprietate.

Scrieți definiția completă a subprogramului.

Exemplu: dacă **n=16** atunci, după apel, **x=2** și **p=4**, dacă **n=216** atunci, după apel, **x=6** și **p=3**, iar dacă **n=12** atunci, după apel, **x=12** și **p=1**. **(10p.)**

```
void Putere(int n, int &x, int &p)
{
    for (p = 30; p >= 2; p--)
    {
       int st = 2, dr = n;

        while (st <= dr)
        {
            int m = (st + dr) / 2;
            int putere = 1;

            for (int i = 1; i <= p && putere <= n; i++)
                putere *= m;

            if (putere == n)
            {
                x = m;
                return;
            }

            if (putere < n)
                st = m + 1;
            else
                dr = m - 1;
        }
    }

    x = n;
    p = 1;
}
```

3.49 Subiect 2023. Varianta SIMULARE

1. Subprogramul **NrImp** are trei parametri:

- **x** și **y**, prin care primeşte câte un număr natural ($2 \le x < y \le 10^9$)
- **nr**, prin care furnizează numărul valorilor naturale din intervalul **[x,y]** cu trei divizori pozitivi impari.

Scrieţi definiţia completă a subprogramului.

Exemplu: dacă **x=4** și **y=50**, după apel **nr=6** (pentru valorile **9**, **18**, **25**, **36**, **49**, **50**). **(1(**

```
void NrImp(int x, int y, int &nr)
{   nr = 0;
    for (int n = x; n <= y; n++)
    {
        int m = n;
        while (m % 2 == 0)
            m /= 2;
        int d1 = sqrt(m);
        if (d1 * d1 == m)
        {
            int ok = 1;
            if (d1 < 2)
                ok = 0;
            else
                for (int d = 2; d * d <= d1; d++)
                    if (d1 % d == 0)
                    {
                        ok = 0;
                        break;
                    }
            if (ok)
                nr++;
        }
    }
}
```

3.50 Subiect 2023. Varianta MODEL

Subprogramul **DoiTrei** are un parametru, **n**, prin care primeşte un număr natural ($n \in [0,10^9]$). Subprogramul returnează valoarea **1** dacă toate cifrele lui **n** sunt din mulțimea **{2,3}** sau valoarea **0** în caz contrar. Scrieți definiția completă a subprogramului.
Exemplu: dacă **n=22323** sau **n=3**, atunci subprogramul returnează **1**, iar dacă **n=2023** atunci subprogramul returnează **0**. **(10p.)**

```
int DoiTrei(int n)
{
    if (n == 0)
        return 0;

    while (n != 0)
    {
        int c = n % 10;
        if (c != 2 && c != 3)
            return 0;
        n /= 10;
    }

    return 1;
}
```

3.51 Subiect 2024. Varianta 8

Un număr natural nenul, **n**, se numește **moderat** dacă este egal cu produsul a două numere prime, iar acestea sunt consecutive în șirul numerelor prime (**2**, **3**, **5**, **7**, **11**, **13**, **17**...).
Subprogramul **moderat** are un singur parametru, **n**, prin care primește un număr natural ($n \in [1, 10^5]$). Subprogramul returnează valoarea **1**, dacă **n** este un număr moderat, sau valoarea **0**, în caz contrar.
Scrieți definiția completă a subprogramului.
Exemplu: dacă **n=35**, subprogramul returnează **1** (**35=5·7**), iar dacă **n=18** sau **n=55** sau **n=4**, subprogramul returnează **0**. **(10p.)**

```
int moderat(int n)
{
    for (int d = 2; d * d <= n; d++)
        if (n % d == 0)
        {
            int q = n / d;
            // verific dacă d este prim
            int prim1 = 1;
            for (int i = 2; i * i <= d; i++)
                if (d % i == 0)
                {
                    prim1 = 0;
                    break;
                }
            // verific dacă q este prim
            int prim2 = 1;
            for (int i = 2; i * i <= q; i++)
                if (q % i == 0)
                {
                    prim2 = 0;
                    break;
                }
            if (prim1 && prim2)
            {
                int ok = 1;
                for (int p = d + 1; p < q; p++)
                {
                    int prim = 1;
                    if (p < 2) prim = 0;
                    for (int i = 2; i * i <= p; i++)
                        if (p % i == 0){ prim = 0; break;}
                    if (prim){ ok = 0; break;}
                }
                if (ok)
                    return 1;
            }
        }
    return 0;
}
```

3.52 Subiect 2024. Varianta 3

Un număr natural se numește **major impar** dacă suma divizorilor săi proprii impari este strict mai mare decât suma divizorilor săi proprii pari. Divizorii proprii ai unui număr sunt divizorii săi naturali diferiți de **1** și de el însuși.
Exemplu: **18** este număr major impar (divizorii săi proprii pari sunt **2**, **6**, cei impari **3**, **9**, iar **3+9>2+6**).
Subprogramul **majImp** are doi parametri, **a** și **b**, prin care primește câte un număr natural ($2 \le a \le b \le 10^4$).
Subprogramul returnează cel mai mic număr major impar din intervalul **[a,b]**, sau valoarea **0**, dacă în interval nu există un astfel de număr. Scrieţi în C/C++ definiţia completă a subprogramului.
Exemplu: dacă **a=16**, **b=30**, atunci subprogramul returnează **18**. **(10p.)**

```
int majImp(int a, int b)
{
    for (int n = a; n <= b; n++)
    {
        int sImp = 0, sPar = 0;
        for (int d = 2; d * d <= n; d++)
            if (n % d == 0)
            {
                if (d != n)
                {
                    if (d % 2 == 0)
                        sPar += d;
                    else
                        sImp += d;
                }
                int q = n / d;
                if (q != d && q != n)
                {
                    if (q % 2 == 0)
                        sPar += q;
                    else
                        sImp += q;
                }
            }
        if (sImp > sPar)
            return n;
    }
    return 0;
}
```

3.53 Subiect 2024. Varianta 4

La un laborator sunt studiate aglomerările de fulgi de nea formate din câte **noua** cristale de **patru** tipuri diferite date (notate cu **1**, **2**, **3** sau **4**), astfel încât din fiecare tip să existe cel puțin câte un cristal. O astfel de aglomerare de fulgi a fost reprezentată printr-un număr natural, în care fiecare cifră reprezintă tipul unui cristal.
Subprogramul **fulg** are un parametru, **n**, prin care primește un număr natural ($n \in [0, 10^9)$). Subprogramul returnează valoarea **1**, dacă prin **n** este reprezentată o aglomerare de fulgi de nea dintre cele studiate, sau **0** în caz contrar. Scrieți în C/C++ definiția completă a subprogramului.
Exemplu: dacă **n=112243413** subprogramul returnează **1**, iar dacă **n=12314** sau **n=112253513** sau **n=112243457** sau **n=111122223**, subprogramul returnează **0**. **(10p.)**

```
int fulg(int n)
{
    int f[5] = {0}, nrCif = 0;

    while (n)
    {
        int c = n % 10;
        if (c < 1 || c > 4)
            return 0;

        f[c]++;
        nrCif++;
        n /= 10;
    }

    if (nrCif != 9)
        return 0;

    for (int i = 1; i <= 4; i++)
        if (f[i] == 0)
            return 0;

    return 1;
}
```

3.54 Subiect 2024. Varianta MODEL

Subprogramul **produs** are doi parametri, **a** și **b**, prin care primește câte un număr natural din intervalul $[1,10^3]$. Subprogramul returnează produsul divizorilor naturali comuni lui **a** și **b**.
Scrieți definiția completă a subprogramului.
Exemplu: dacă **a=20** și **b=12**, atunci subprogramul returnează valoarea 8 ($1 \cdot 2 \cdot 4=8$). **(10p.)**

```
int produs(int a, int b)
{
    int x = a, y = b;

    // Calcul CMMDC
    while (y != 0)
    {
        int r = x % y;
        x = y;
        y = r;
    }
    //produsul divizorilor lui CMMDC
    int p = 1;
    for (int d = 1; d <= x; d++)
        if (x % d == 0)
            p *= d;

    return p;
}
```

3.55 Subiect 2025. Varianta 4

Un fermier are un teren de formă dreptunghiulară, pe care vrea să îl împartă în parcele de arie egală, astfel încât numărul de parcele și aria să fie valori naturale, iar numărul de parcele să fie par și strict mai mic decât aria unei parcele. Subprogramul **`teren`** are doi parametri, **`x`** și **`y`**, prin care primește două numere naturale din intervalul **`[2,100]`**, reprezentând lungimea și lățimea terenului, măsurate în metri. Subprogramul afișează toate soluțiile posibile de împărțire a terenului, fiecare soluție pe câte o linie a ecranului, sub forma a două numere, separate prin mesajul **`parcele de arie`** și spații adecvate, unde primul număr reprezintă numărul de parcele obținute, iar al doilea aria unei parcele corespunzătoare, ca în exemplu. Dacă nu există nicio astfel de soluție, subprogramul afișează pe ecran mesajul **`nu exista`**. Scrieți definiția completă a subprogramului C/C++.

Exemplu: pentru **`x=6`** și **`y=8`**, subprogramul afișează pe ecran valorile alăturate, nu neapărat în această ordine (terenul poate fi împărțit și în **`3`** parcele de arie **`16`** sau **`8`** parcele de arie **`6`**, dar nu corespund cerinței).**(10p.)**

```
2 parcele de arie 24
4 parcele de arie 12
6 parcele de arie 8
```

```
void teren(int x, int y)
{
    int p = x * y;
    bool ok = false;

    for (int n = 2; n <= p; n += 2)
        if (p % n == 0)
        {
            int aria =  p/ n;
            if (n < aria)
            {
                cout << n << " parcele de arie " << aria << endl;
                ok = true;
            }
        }

    if (ok==0)
        cout << "nu exista";
}
```

3.56 Subiect 2025. Varianta 1

Numărul natural **an** este **ascendent** al numărului natural **n**, dacă oricare dintre cifrele lui **an** este mai mare sau egală cu cifra unităților lui **n**.
Exemplu: oricare dintre numerele **7**, **9**, **98** sau **7998** este ascendent al lui **827**, dar numărul **857** nu este ascendent al lui **827**.
Subprogramul **ascendent** are trei parametri:
- **n**, prin care primește un număr natural (**n**$\in$**[0,10³)**);
- **x** și **y**, prin care primește câte un număr natural din intervalul **[0,10³)** (**x<y**).

Subprogramul returnează suma ascendenților lui **n** din intervalul **[x,y]**, sau valoarea **0**, dacă nu există niciun astfel de ascendent. Scrieți definiția completă a subprogramului C/C++.
Exemplu: dacă **n=827**, **x=9**, **y=800**, subprogramul returnează **7893** (**9+77+78+79+87+88+89+97+98+99+777+778+779+787+788+789+797+798+799=7893**). **(10p.)**

```
int ascendent(int n, int x, int y)
{
    int c = n % 10, sum = 0;
    for (int i = x; i <= y; i++)
    {
        int t = i;
        bool ok = true;

        if (t == 0)
        {
            if (0 < c)
                ok = false;
        }
        else
            while (t != 0 && ok)
            {
                if (t % 10 < c)
                    ok = false;
                t /= 10;
            }
        if (ok)
            sum = sum + i;
    }
    return sum;
}
```

3.57 Subiect 2025. Varianta 6

Subprogramul **diviz** are un singur parametru, **n**, prin care primește un număr natural ($n \in [1, 10^9]$). Subprogramul returnează cel mai mare divizor al lui **n** care este pătrat perfect.
Scrieți definiția completă a subprogramului C/C++.
Exemplu: pentru **n=72** subprogramul returnează **36**, pentru **n=16** subprogramul returnează **16**, iar pentru **n=15** subprogramul returnează **1**. **(10p.)**

```
int diviz(int n)
{
    for (int d = n; d >= 1; d--)
        if (n % d == 0)
        {
            int rad = (int)sqrt(d);
            if (rad * rad == d)
                return d;
        }
    return 1;
}
```

3.58 Subiect 2025. Varianta SIMULARE

Un număr „care aduce bucurie” - **harsad** (sau număr Niven), este un număr întreg divizibil cu suma cifrelor sale.
Subprogramul **harsad** are doi parametri:

- **k**, prin care primește un număr natural ($k \in [1, 10^6]$);
- **n**, prin care furnizează cel mai mare număr natural harsad mai mic sau egal cu **k**.

Scrieți definiția completă a subprogramului.
Exemplu: pentru **k=2027**, după apel, **n=2025** (**2+0+2+5=9**, iar **2025** este divizibil cu **9**). **(10p.)**

```
void harsad(int k, int &n)
{
    while (k >= 1)
    {
        int s = 0, x = k;
        while (x != 0)
        {
            s += x % 10;
            x /= 10;
        }
        if (k % s == 0)
        {
            n = k;
            break;
        }
        k--;
    }
    if (k == 0) n = 0;
}
```

3.59 Subiect 2025. Varianta MODEL

Două numere se numesc **oglindite** dacă fiecare se obține din celălalt, prin parcurgerea cifrelor acestuia de la dreapta la stânga. Două numere se numesc **par-oglindite** dacă numerele obținute din acestea, prin îndepărtarea tuturor cifrelor lor impare sau nule, sunt oglindite.
Subprogramul **pao** are trei parametri:

- **x** și **y**, prin care primește câte un număr natural din intervalul $[0,10^9]$;
- **rez**, prin care furnizează valoarea **1**, dacă **x** și **y** sunt par-oglindite, sau valoarea **0**, în caz contrar.

Scrieți definiția completă a subprogramului.
Exemplu: dacă **x=814** și **y=7003485**, sau **x=14** și **y=70345**, după apel **rez=1**,
iar dacă **x=814** și **y=7003465**, sau **x=814** și **y=7003845**, sau **x=15** și **y=510**, după apel **rez=0**. **(10p.)**

```
void pao(int x, int y, int &rez)
{
    int a = 0, p = 1, cif, b=0;

    while (x != 0)
    {
        cif = x % 10;
        if (cif != 0 && cif % 2 == 0)
        {
            a = cif * p + a;
            p *= 10;
        }
        x /= 10;
    }
    while (y != 0)
    {
        cif = y % 10;
        if (cif != 0 && cif % 2 == 0)
            b = b * 10 + cif;
        y /= 10;
    }

    if (a == b)
        rez = 1;
    else
        rez = 0;
}
```

3.60 Subiect 2026. Varianta 3

Numim **numar uniform** asociat unei valori naturale numărul obținut din aceasta prin eliminarea fie a tuturor cifrelor sale pare, fie a tuturor cifrelor sale impare.
Exemplu: lui **19472** i se asociază numerele uniforme **197** și **42**.
Subprogramul **ImparPar** are un singur parametru, **n**, prin care primește un număr natural cu toate cifrele nenule ($n \in [11, 10^9)$), având cel puțin o cifră pară și cel puțin o cifră impară. Subprogramul returnează un număr obținut din cifrele numărului uniform impar asociat lui **n**, urmate de cifrele numărului uniform par asociat lui **n**, ca în exemplu. Scrieți definiția completă a subprogramului.
Exemplu: dacă **n=19472**, subprogramul returnează numărul **19742**. **(10p.)**

```
int ImparPar(int n)
{
    int imp = 0, par = 0, p_10 = 1, i_10 = 1, x = n;
    while (x != 0)
    {
        int c = x % 10;
        if (c % 2 == 0)
        {
            par = c * p_10 + par;
            p_10 *= 10;
        }
        else
        {
            imp = c * i_10 + imp;
            i_10 *= 10;
        }
        x /= 10;
    }
    return imp * p_10 + par;
}
```

3.61 Subiect 2026. Varianta 5

O editură realizează seturi turistice, formate din câte o broșură de prezentare a zonei locale și câte un catalog al producătorilor locali, astfel încât dacă o broșură are **p** pagini, atunci **p** este un număr prim, iar catalogul din același set are **3·p** pagini.
Subprogramul **pagini** are un parametru, **n**, prin care primește un număr natural ($n \in [2, 10^4]$). Subprogramul returnează valoarea **1** dacă, folosind toate cele **n** pagini, se poate realiza un set turistic, sau valoarea **0** în caz contrar. Scrieți în C/C++ definiția completă a subprogramului.
Exemplu: dacă **n=44**, subprogramul returnează valoarea **1** (se poate obține o broșură de **11** pagini și un catalog de **33=3·11** pagini), iar dacă **n=48** sau **n=27**, subprogramul returnează **0**. **(10p.)**

```
int pagini(int n)
{
     if (n % 4 != 0)
          return 0;
     int p = n / 4;
     if (p < 2)
          return 0;
     for (int d = 2; d * d <= p; d++)
          if (p % d == 0)
               return 0;
     return 1;
}
```

3.62 Subiect 2026. Varianta 4

La o paradă de modă se prezintă modele de rochii și costume, iar ordinea acestora este dată ca o succesiune de cifre, cele impare reprezentând rochii, iar cele pare costume; în prezentare este inclus cel puțin un costum.

Subprogramul **moda** are doi parametri:

- **n**, prin care primește un număr natural din intervalul **[0,10⁹]**, ale cărui cifre, de la stânga la dreapta, corespund modelelor, în ordinea prezentării lor;
- **pc**, prin care furnizează numărul de ordine al primului costum prezentat.

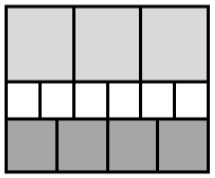
Scrieți definiția completă a subprogramului C/C++.

Exemplu: dacă **n=576798** sau **n=5700**, atunci **pc=3**. **(10p.)**

```
void moda(int n, int &pc)
{
     while (n != 0)
     {
          if (n%2 == 0)
               pc = 0;
          else
               pc++;
          n /= 10;

     }
     pc++;
}
```

3.63 Subiect 2026. Varianta SIMULARE

Într-un depozit se montează un raft cu trei polițe de aceeași lungime și de lățimi **x**, **y** respectiv **z**. Pe fiecare poliță se plasează, una lângă alta, cutii de forma unui cub cu latura egală cu lățimea poliței. În depozit există suficiente cutii de fiecare tip, iar raftul este echilibrat doar dacă fiecare poliță este ocupată complet de cutii.
Subprogramul **depozit** are trei parametri, **x**, **y** și **z**, prin care primește trei numere naturale din intervalul [10,100], reprezentând lățimile celor trei polițe, exprimate în centimetri. Subprogramul returnează un număr natural, reprezentând lungimea minimă, exprimată în centimetri, a unei polițe a raftului echilibrat.
Scrieți definiția completă a subprogramului C/C++.
Exemplu: pentru **x=40**, **y=20** și **z=30**, subprogramul returnează valoarea **120** (raftul are trei polițe, fiecare cu lungime de **120 cm**: prima poate conține **3** cutii cu latura de **40 cm**, a doua **6** cutii cu latura de **20 cm**, iar a treia **4** cutii cu latura de **30 cm**). **(10p.)**



```
int depozit(int x, int y, int z)
{
    int m = min(x,y);
    m = min(m,z);
    int i = m;
    while (1)
    {
        if (i%x == 0 && i%y == 0 && i%z == 0)
            return i;
        i = i + m;
    }
}
```

3.64 Subiect 2026. Varianta MODEL

Subprogramul **Plus** are un singur parametru, **n**, prin care primește un număr natural ($n \in [10, 10^9]$). Subprogramul înlocuiește în **n** fiecare secvență **25** cu câte o secvență **26** și furnizează, prin același parametru, numărul obținut. Dacă nu se înlocuiește nicio secvență, subprogramul furnizează numărul nemodificat.
Scrieți definiția completă a subprogramului C/C++.
Exemplu: dacă **n=202535250**, după apel **n=202635260**. **(10p.)**

```
void Plus(int &n)
{
    int r = 0, p = 1;

    while (n > 0)
    {
        int u = n % 10;
        n /= 10;
        if (u == 5 && n % 10 == 2)
            u = 6;
        r = r + u * p;
        p *= 10;
    }
    n = r;
}
```