Cuprins

Introducere

Subiectul III al unei variante de bacalaureat la informatică urmărește abilitatea elevilor de a aplica, combina, îmbina anumiți algoritmi elementari pe diverse situații, fenomene descrise de cele mai multe ori matematic. Dacă primele 2 probleme ale acestui subiect dau posibilitatea elevului să implementeze diverse soluții, elevii chiar găsesc rezolvări deosebit de creative, ultima problemă trebuie abordată diferit deoarece majoritatea acestor cerințe impun limite ce se referă la spațiul de memorare și timp. Pentru o rezolvare cu punctaj maxim, elevul trebuie să gândească diferit față de alte implementări și să analizeze cu mare atenție datele ce intervin în prelucrare. În același timp, un aspect deosebit de important îl constituie citirea cu atenție a cerinței și extragerea din text al tuturor elementelor ce pot influența în mod pozitiv maniera de rezolvare și corectitudinea rezultatului.

Propun în prezentul auxiliar rezolvarea unor teste de bacalaureat cu evidențierea unor aspecte mai importante prin comentarea acelor rânduri.

Orice altă soluție a unei probleme este validă și bună de luat în considerare dacă respectă întocmai cerințele impuse.

Testul de antrenament nr. 12

https://www.pbinfo.ro/resurse/9dc152/examene/2021/antrenament/E_d_Informatica_2021_sp_MI_C_Test_12.pdf

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.)**

```
//prelucrari de cifre
void frate(int x, int &y)//prin y furnizeaza
{
   y=0;//esential
   int ok=0;//pp ca putem construi pe y
   int p=1;//construim pe y direct
   while(x&&ok==0)
   {
      if(x%10==9)ok=1;//NU putem construi pe y
      else
      {
         y=p*(x%10+1)+y;
         p=p*10;
      }
      x=x/10;
   }
   if(ok==1)y=-1;
}
```

Scrieți un program C/C++ care citește de la tastatură numere naturale: **n** (**n∈[2,20]**), **k** (**k∈[1,n]**) și **n·n** numere din intervalul **[0,10⁹]**, elemente ale unui tablou bidimensional cu **n** linii și **n** coloane. Programul transformă tabloul în memorie, deplasând circular spre dreapta, cu câte o poziție, toate elementele situate pe linia a **k**-a, în stânga diagonalei secundare, ca în exemplu. Elementele tabloului obținut sunt afișate pe ecran, fiecare linie pe câte o linie a ecranului, cu elementele fiecărei linii separate prin câte un spațiu.

Exemplu: pentru **n=5**, **k=2** și tabloul

```
2 3 4 5 6
2 4 6 8 0
7 8 9 0 1
3 5 7 9 1
7 3 8 5 6
```

se obține tabloul

```
2 3 4 5 6
6 2 4 8 0
7 8 9 0 1
3 5 7 9 1
7 3 8 5 6
```

(10p.)

```
#include <iostream>
using namespace std;
int main()
{
   int a[21][21],n,k,i,j;
   cin>>n>>k;
   for(i=1;i<=n;i++)
      for(j=1;j<=n;j++)
         cin>>a[i][j];
   //suficient sa parcurg linia k pana la diagonala secundara
   //deoarece permutam circular spre dreapta, scriem for de la ultima
colona de prelucrat spre prima
   //i+j==n+1 pe diagonala secundara => j=n+1-i, si desigur i=k
   int aux=a[k][n+1-k-1];
   for(j=n+1-k-1;j>=2;j--)
      a[k][j]=a[k][j-1];
   a[k][1]=aux;
   //afisare matrice
   for(i=1;i<=n;i++)
   {
      for(j=1;j<=n;j++)
         cout<<a[i][j]<<" ";
      cout<<endl;
   }
   return 0;
}
```

Fișierul **bac.txt** conține un șir de cel mult 10^5 numere naturale din intervalul $[1,10^9]$, separate prin câte un spațiu. Se cere să se afișeze pe ecran cea mai mare poziție pe care ar putea-o ocupa primul termen al șirului aflat în fișier în șirul format cu aceleași valori, ordonat descrescător. Proiectați un algoritm eficient din punctul de vedere al memoriei utilizate și al timpului de executare.
Exemplu: dacă fișierul **bac.txt** conține numerele **15 7 15 17 6 4 21**
se afișează pe ecran **4** (**15** se află pe a treia și pe a patra poziție în șirul **21**, **17**, **15**, **15**, **7**, **6**, **4**).
a. Descrieți în limbaj natural algoritmul proiectat, justificând eficiența acestuia. **(2p.)**
b. Scrieți programul C/C++ corespunzător algoritmului proiectat. **(8p.)**

```
#include <iostream>
#include <fstream>//esential
using namespace std;
int main()
{
   ifstream f("bac.txt");
   int x,y,k;
   //e suficient sa numaram cate numere din fisiser sunt mai mari sau
egale cu primul
   f>>x;//citim pe primul
   k=1;//si x ocupa o pozitie in cadrul sirului
   while(f>>y)//citim pe rand celelalte numere
      if(x<=y)k++;
   cout<<k;
   //recomandare
   f.close();
   return 0;
}
```

Testul de antrenament nr. 11

https://www.pbinfo.ro/resurse/9dc152/examene/2021/antrenament/E_d_Informatica_2021_sp_MI_C_Test_11.pdf1.1.

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.)**

```
//prelucrari de cifre
void imog(int x, int y, int &rez)//prin rez funizeaza
{
   int xx=0,p=1;//construim xx direct cu cifre impare din x
   while(x)
   {
      if(x%2==1)
      {
         xx=p*(x%10)+xx;
         p=p*10;
      }
      x=x/10;
   }
   int yy;//construim pe yy cu cifre impare invers din y
   while(y)
   {
      if(y%2==1)
         yy=yy*10+y%10;
      y=y/10;
   }
   //test final
   if(xx==yy&&xx!=0)rez=1;//obligatoriu egale si nu egale cu 0
   else rez=0;
}
```

Scrieți un program C/C++ care citește de la tastatură numere naturale: **n** (**n∈[2,20]**), **k** (**k∈[2,n]**) și **n·n** numere din intervalul **[0,10⁹]**, elemente ale unui tablou bidimensional cu **n** linii și **n** coloane. Programul transformă tabloul în memorie, interschimbând șirul elementelor situate pe linia a **k**-a, în stânga diagonalei principale, parcurse de la stânga la dreapta, cu șirul elementelor situate pe coloana a **k**-a, deasupra diagonalei principale, parcurse de sus în jos, ca în exemplu. Elementele tabloului obținut sunt afișate pe ecran, fiecare linie pe câte o linie a ecranului, cu elementele fiecărei linii separate prin câte un spațiu.

Exemplu: pentru **n=5**, **k=4** și tabloul

2	4	3	5	6
8	0	9	8	7
2	6	9	0	5
6	1	3	6	9
7	3	9	4	2

se obține tabloul

2	4	3	6	6
8	0	9	1	7
2	6	9	3	5
5	8	0	6	9
7	3	9	4	2

(10p.)

```
#include <iostream>
using namespace std;
int main()
{
   int a[21][21],n,k,i,j;
   cin>>n>>k;
   for(i=1;i<=n;i++)
      for(j=1;j<=n;j++)
         cin>>a[i][j];
   //observam interschimbul a[i][j]<->a[j][i]
   //e suficient sa parcurgem linia k pana la diagonala principala
(exclusiv)
   for(j=1;j<k;j++)
   {
      int aux=a[k][j];//interschimb
      a[k][j]=a[j][k];
      a[j][k]=aux;
   }
   for(i=1;i<=n;i++)
   {
      for(j=1;j<=n;j++)
         cout<<a[i][j]<<" ";
      cout<<endl;
   }
   return 0;
}
```

Se consideră șirul **1**, **3**, **7**, **13**, **21**, **31**, **43** **...** definit astfel: $f_0=1$, iar $f_n=f_{n-1}+2\cdot n$, dacă $n\geq 1$ (unde **n** este un număr natural).
Se citesc de la tastatură două numere naturale din intervalul $[1,10^9]$, **x** și **y** (**x<y**), reprezentând doi termeni aflați pe poziții consecutive în șirul dat, și se cere să se scrie în fișierul text **bac.out**, separați prin câte un spațiu, toți termenii șirului mai mici sau egali cu **y**, în ordine inversă a apariției lor în șir. Proiectați un algoritm eficient din punctul de vedere al spațiului de memorie și al timpului de executare.
Exemplu: dacă **x=21** și **y=31**, fişierul conţine valorile
31 21 13 7 3 1
a. Descrieți în limbaj natural algoritmul proiectat, justificând eficiența acestuia. **(2p.)**
b. Scrieți programul C/C++ corespunzător algoritmului proiectat. **(8p.)**

```
#include <iostream>
#include <fstream>//esential
using namespace std;
int main()
{
   ofstream f("bac.out");//atentie
   int x,y;
   cin>>x>>y;
   int n=(y-x)/2;//calculam pe n
   f<<y<<" "<<x<<" ";//scriem primele 2 valori, le stim
   for(int i = n-1; i >= 1; i--)
   {
      x=x-2*i;//reactualizam pe x, extras din formula
      f<<x<<" ";//afisare
   }
   return 0;
}
```

Testul de antrenament nr. 10

https://www.pbinfo.ro/resurse/9dc152/examene/2021/antrenament/E_d_Informatica_2021_sp_MI_C_Test_10.pdf

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** și **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.)**

```
//divizibilitate
int armonie(int x, int y)//nu se specifica variabila returnata
{
   int sx=0;//determin suma divizorilor lui x
   for(int d=1;d<=x;d++)
      if(x%d==0)sx=sx+d;
   int sy=0;//determin suma divizorilor lui y
   for(int d=1;d<=y;d++)
      if(y%d==0)sy=sy+d;
   //direct return, expresia se evalueaza logic
   return ((x+y>sx&&x+y<sy)||(x+y>sy&&x+y<sx));
   //atentie nu stim care suma e mai mare
}
```

O valoare **filtrează** două șiruri dacă există doi termeni care au acea valoare, unul fiind în primul șir, iar celălalt în al doilea șir.
Scrieți un program C/C++ care citește de la tastatură numere naturale din intervalul `[2,20]`: `m`, `n` și elementele unui tablou bidimensional cu `m` linii și `n` coloane, cu proprietatea că nu există două elemente egale situate pe aceeași linie sau pe aceeași coloană.

Programul afișează pe ecran valorile care pot filtra șirul format din primele `n-1` elemente de pe prima linie, respectiv șirul format din ultimele `m-1` elemente ale ultimei coloane a tabloului, ca în exemplu. Valorile sunt afișate într-o ordine oarecare, separate prin câte un spațiu, sau mesajul `nu exista`, dacă nu există astfel de valori.
Exemplu: pentru `m=5`, `n=4` și tabloul alăturat, se afișează pe ecran, nu neapărat în această ordine, numerele `4 7` **(10p.)**

4	5	**7**	2
2	7	3	6
7	6	4	0
6	9	8	**7**
8	0	5	**4**

```
#include <iostream>
using namespace std;
int main()
{
   int a[21][21],n,m,i,j;
   cin>>m>>n;
   for(i=1;i<=m;i++)//m linii
      for(j=1;j<=n;j++)//n coloane
         cin>>a[i][j];
   //parcurg prima linie, fara acel element din colt
   //daca il gasesc pe ultima coloana, il afisez
   //atentie cazul nu exista
   int ok=0;//pp ca NU afisez nimic
   for(j=1;j<=n-1;j++)
   {
      //exista a[1][j] pe ultima coloana?
      int oki=0;//pp ca NU exista
      for(i=2;i<=m;i++)
         if(a[1][j]==a[i][n])
         {
            oki=1;//am gasit
            ok=1;//marchez ca voi face afisare
         }
      //atentie, doar dupa ce termin verificarea coloanei testez
      if(oki==1)cout<<a[1][j]<<" ";
   }
   //doar la final
   if(ok==0)cout<<"nu exista";
   return 0;
}
```

Fişierul **bac.txt** conține un șir de cel mult 10^5 numere naturale din intervalul $[1,10^9]$, separate prin câte un spaţiu.
Se cere să se afişeze pe ecran cea mai mică poziţie pe care ar putea-o ocupa primul termen al șirului aflat în fişier în şirul format cu aceleași valori, ordonat crescător. Proiectați un algoritm eficient din punctul de vedere al memoriei utilizate şi al timpului de executare.
Exemplu: dacă fişierul conţine numerele **15 7 15 17 6 4**
se afişează pe ecran **4** (**15** se află pe a patra și pe a cincea poziție în şirul **4**, **6**, **7**, **15**, **15**, **17**).
a. Descrieți în limbaj natural algoritmul proiectat, justificând eficiența acestuia. **(2p.)**
b. Scrieți programul C/C++ corespunzător algoritmului proiectat. **(8p.)**

```
#include <iostream>
#include <fstream>//esential
using namespace std;
int main()
{
   ifstream f("bac.txt");
   int x,k,y;
   f>>x;//citesc primul numar din fisier
   k=1;//x ocupa deja un loc, voi numara mai departe cate numere
din fisiser sunt mai mici decat x
   while(f>>y)
      if(y<x)k++;
   cout<<k;
   f.close();
   return 0;
}
```

Testul de antrenament nr. 9

https://www.pbinfo.ro/resurse/9dc152/examene/2021/antrenament/E_d_Informatica_2021_sp_MI_C_Test_09.pdf

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.)**

```
#include <iostream>
using namespace std;
//divizibilitate
void divizor(int a, int b, int k, int &nr)//se specifica variabila
returnata
{
   nr=0;//este o problema de numaratoare
   //nu uitam initializarea aici a lui nr
   for(int i=a;i<=b;i++)//iau fiecare numar din intervalul [a,b]
      if(i%k==0&&i%10==k)//din textul problemei
         nr++;
}
int main()
{
   //testare simpla
   int nr;//e suficient declarat acest nr
   divizor(3,50,4,nr);//apel
   cout<<nr;//apoi afisare rezultat
   return 0;
}
```

O valoare **filtrează** două șiruri dacă există doi termeni care au acea valoare, unul fiind în primul șir, iar celălalt în al doilea șir.
Scrieți un program C/C++ care citește de la tastatură numere naturale din intervalul `[2,20]`: `m`, `n` și elementele unui tablou bidimensional cu `m` linii și `n` coloane, cu proprietatea că nu există două elemente egale situate pe aceeași linie sau pe aceeași coloană.

Programul afișează pe ecran valorile care pot filtra șirul format din primele `n-1` elemente de pe prima linie, respectiv șirul format din ultimele `m-1` elemente ale ultimei coloane a tabloului, ca în exemplu. Valorile sunt afișate într-o ordine oarecare, separate prin câte un spațiu, sau mesajul `nu exista`, dacă nu există astfel de valori.
Exemplu: pentru `m=5`, `n=4` și tabloul alăturat, se afișează pe ecran, nu neapărat în această ordine, numerele `4 7` **(10p.)**

4	5	**7**	2
2	7	3	6
7	6	4	0
6	9	8	**7**
8	0	5	**4**

```
#include <iostream>
using namespace std;
int main()
{
   int a[21][21],n,m,i,j;
   cin>>m>>n;
   for(i=1;i<=m;i++)//m linii
      for(j=1;j<=n;j++)//n coloane
         cin>>a[i][j];
   //parcurg prima linie, fara acel element din colt
   //daca il gasesc pe ultima coloana, il afisez
   //atentie cazul nu exista
   int ok=0;//pp ca NU afisez nimic
   for(j=1;j<=n-1;j++)
   {
      //exista a[1][j] pe ultima coloana?
      int oki=0;//pp ca NU exista
      for(i=2;i<=m;i++)
         if(a[1][j]==a[i][n])
         {
            oki=1;//am gasit
            ok=1;//marchez ca voi face afisare
         }
      //atentie, doar dupa ce termin verificarea coloanei testez
      if(oki==1)cout<<a[1][j]<<" ";
   }
   //doar la final
   if(ok==0)cout<<"nu exista";
   return 0;
}
```

Fişierul **bac.txt** conține un șir de cel mult 10^5 numere naturale din intervalul $[1,10^9]$, separate prin câte un spaţiu.
Se cere să se afişeze pe ecran cea mai mică poziţie pe care ar putea-o ocupa primul termen al șirului aflat în fişier în şirul format cu aceleași valori, ordonat crescător. Proiectați un algoritm eficient din punctul de vedere al memoriei utilizate şi al timpului de executare.
Exemplu: dacă fişierul conţine numerele **15 7 15 17 6 4**
se afişează pe ecran **4** (**15** se află pe a patra și pe a cincea poziție în şirul **4**, **6**, **7**, **15**, **15**, **17**).
a. Descrieți în limbaj natural algoritmul proiectat, justificând eficiența acestuia. **(2p.)**
b. Scrieți programul C/C++ corespunzător algoritmului proiectat. **(8p.)**

```
#include <iostream>
#include <fstream>//esential
using namespace std;
int main()
{
   ifstream f("bac.txt");
   int x,k,y;
   f>>x;//citesc primul numar din fisier
   k=1;//x ocupa deja un loc, voi numara mai departe cate numere
din fisiser sunt mai mici decat x
   while(f>>y)
      if(y<x)k++;
   cout<<k;
   f.close();
   return 0;
}
```

Testul de antrenament nr. 8

https://www.pbinfo.ro/resurse/9dc152/examene/2021/antrenament/E_d_Informatica_2021_sp_MI_C_Test_08.pdf

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.)**

```
//divizibilitate, determinare maxim
void nrfp(int n, int &m)//se specifica variabila returnata
{
   int maxim=0;//observam o problema de maxim
   for(int i=2;i<=n;i++)//parcurgem intreg intervalul [2,n]
   {
      //cati factori primi are i?
      int are=0;//pp ca i are 0 factori primi
      int aux=i;//descompunerea in factori primi ai lui i
      //aux va deveni 1 asa ca prelcurez pe aux nu i care e contor
      int d=2;//primul posibil divizor
      while(aux>1)
      {
         int e=0;//pp ca nu se divide la d
         while(aux%d==0)//cat timp pot descompune
         {
            aux=aux/d;//descompun
            e++;//numar, determin puterea
         }
         if(e>0)//am descompus
            are++;//ii numar factorul prim
         d++;//pregatesc urmatorul posibil divizor
      }
      //atentie, ca sa memorez ultimul (cel mai mare din interval)
      //punem => ca sa suprascriu pe m
      if(are>=maxim)//daca intalnesc un numar mai mare de divizori
      {
         maxim=are;//memorez maximul
         m=i;//memorez acel i cu cei mai multi divizori
      }
   }
}
```

Scrieți un program C/C++ care citește de la tastatură numărul natural `n` (`n∈[5,50]`) și elementele unui tablou bidimensional cu `n` linii și `n` coloane, numere naturale din intervalul `[0,10²]`. Programul afișează pe ecran suma numerelor din zona delimitată de cele două diagonale și ultima coloană a tabloului, ca în exemplu.
Exemplu: pentru `n=7` și tabloul alăturat, se afișează pe ecran `12`.
(10p.)

1	2	3	4	5	5	6
7	8	9	0	3	1	2
4	6	8	0	1	1	3
8	6	3	6	2	4	7
5	7	9	2	2	5	8
1	4	7	0	5	3	6
9	2	5	8	5	9	1

```
#include <iostream>
using namespace std;
//matrice patratica, parcurgere, suma
int main()
{
   int a[51][51],n,i,j,s;
   cin>>n;
   for(i=1;i<=n;i++)
      for(j=1;j<=n;j++)
         cin>>a[i][j];
   s=0;//urmeaza sa insumez elemente
   //parcurg matricea, atentie la elementele...
   for(i=1;i<=n;i++)
      for(j=1;j<=n;j++)
      //deasupra diagonalei principale
      //si sub diagonala secundara
      //si nu pe ultima coloana
         if((i<j)&&(i+j>n+1)&&(j!=n))
            s=s+a[i][j];
   cout<<s;
   return 0;
}
```

Fișierul **bac.txt** conține un șir de cel mult 10^6 numere naturale din intervalul $[0,10^9]$.
Se cere să se determine și să se afișeze pe ecran, separate printr-un spațiu, ultimele două numere impare (nu neapărat distincte) din șirul aflat în fișier, sau mesajul **nu exista**, dacă nu există două astfel de numere. Proiectați un algoritm eficient din punctul de vedere al memoriei utilizate și al timpului de executare.
Exemplu: dacă fișierul conține valorile **122 1635 628 1413 1647 900 3001 4252**
se afișează pe ecran **1647 3001**
a. Descrieți în limbaj natural algoritmul proiectat, justificând eficiența acestuia. **(2p.)**
b. Scrieți programul C/C++ corespunzător algoritmului proiectat. **(8p.)**

```
#include <iostream>
#include <fstream>//esential
using namespace std;
int main()
{
   ifstream f("bac.txt");
   int x,imp1,imp2;
   imp1=imp2=0;//initializam cele 2 cu valori pare
   while(f>>x)//citesc cate un x din fisiser
      if(x%2==1)//daca este impar
      {
         //sunt atent la acest transfer de valori
         imp1=imp2;//primul devine al doilea
         imp2=x;//al doilea devine cel curent citit impar
      }
   if(imp2==0)//nu am avut suficiente
      cout<<"nu exista";
   else cout<<imp1<<" "<<imp2;
   //recomandare
   f.close();
   return 0;
}
```

Testul de antrenament nr. 7

https://www.pbinfo.ro/resurse/9dc152/examene/2021/antrenament/E_d_Informatica_2021_sp_MI_C_Test_07.pdf

Subprogramul **afisare** are trei parametri:

- **x** și **y**, prin care primește câte un număr natural din intervalul $[0,10^6]$ ($x \leq y$);
- **k**, prin care primește un număr natural ($k \in [2,10^2]$).

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.)**

```
//functie de afisare
void afisare(int x, int y, int k)
{
   int afisare=0;//nu am inca nicio afisare
   for(int i=x;i<=y;i++)//parcurg intervalui [x,y]
   {
      cout<<i<<" ";
      afisare++;//numar afisarea
      if(afisare%k==0)//la multiplu de k
         cout<<"* ";//pun steluta
   }
   //atentie sa nu dublez ultima *
   if(afisare%k!=0)
      cout<<"*";
}
```

Scrieți un program C/C++ care citește de la tastatură un număr natural, **x** (**x∈[1,10⁹]**), și construiește în memorie un tablou bidimensional, pentru care atât numărul de linii, cât și numărul de coloane sunt egale cu numărul de cifre ale lui **x**, iar elementele fiecărei linii au ca valori cifrele lui **x**, în ordine, ca în exemplu.

Elementele tabloului obținut sunt afișate pe ecran, linie cu linie, fiecare linie a tabloului pe câte o linie a ecranului, cu elementele de pe aceeași linie separate prin câte un spațiu.
Exemplu: dacă **x=1359**, se afișează tabloul alăturat. **(10p.)**

```
1 3 5 9
1 3 5 9
1 3 5 9
1 3 5 9
```

```
#include <iostream>
using namespace std;
//generare de matrice
int main()
{
   int a[10][10],n,i,j,x;
   cin>>x;
   //matricea va avea dimensiunea n = cate cifre are x
   n=0;
   int aux=x;//mai am nevoie de x
   while(aux)
   {
      n++;
      aux=aux/10;
   }
   //imi convine sa parcurg pe coloane de la ultima spre prima
   for(j=n;j>=1;j--)
   {
      for(i=1;i<=n;i++)
            a[i][j]=x%10;//pun ultima cifra pe coloana j
      x=x/10;//o tai
   }

   //afisare matrice
   for(i=1;i<=n;i++)
   {
      for(j=1;j<=n;j++)
         cout<<a[i][j]<<" ";
      cout<<endl;
   }
   return 0;
}
```

Fișierul **bac.txt** conține cel mult 10^6 cifre, separate prin câte un spațiu.
Se cere să se afișeze pe ecran, separate prin câte un spațiu, toate cifrele pare care apar în fișier sau mesajul **nu exista**, dacă nu există astfel de cifre. Proiectați un algoritm eficient din punctul de vedere al timpului de executare.
Exemplu: dacă fișierul conține cifrele `3 3 0 8 2 1 2 1 3 7 1 5 2 7 1 0 3 2 3`
pe ecran se afișează, de exemplu în ordine crescătoare, cifrele `0 0 2 2 2 2 8`
a. Descrieți în limbaj natural algoritmul proiectat, justificând eficiența acestuia. **(2p.)**
b. Scrieți programul C/C++corespunzător algoritmului proiectat. **(8p.)**

```
#include <iostream>
#include <fstream>//esential
using namespace std;
//observam prelucrare de cifre, afisare crescatoare
//impreuna cu numarul acestora
//=> vector de frecventa
int main()
{
   ifstream f("bac.txt");
   int v[10]={0};//Pas1
   int x;
   while(f>>x)//citesc cate un x din fisiser
      v[x]++;//Pas2
   //Pas3
   int ok=0;//cazul nu exista
   for(x=0;x<=8;x=x+2)//doar cele pare ne intereseaza
      for(int i=1;i<=v[x];i++)
      {
         cout<<x<<" ";
         ok=1;//marchez ca am afisat
      }
   if(ok==0)//daca nu am afisat nimic
      cout<<"nu exista";
   //recomandare
   f.close();
   return 0;
}
```

Testul de antrenament nr. 6

https://www.pbinfo.ro/resurse/9dc152/examene/2021/antrenament/E_d_Informatica_2021_sp_MI_C_Test_06.pdf

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.)**

```
//prelucrare cifre
void numar(int n, int c, int &m)//se specifica variabila
{
   //elimin cifre
   //aleg constructie directa
   //construiesc un numar m fara cifra c
   m=0;//important
   int p=1;
   if(n==0&&c==0)//cazul n=0 si toate cifrele egale cu c
      m=-1;
   int ok=0;//pp ca nu pun nicio cifra in m
   while(n)
   {
      if(n%10!=c)//am nevoie de aceasta cifra
      {
         m=p*(n%10)+m;
         p=p*10;
         ok=1;//am pus cifra in m
      }
      n=n/10;
   }
   if(ok==0)//nu am gasit cifre diferite de c
      m=-1;
}
```

Scrieți un program C/C++ care citește de la tastatură un număr natural, **n** (**n∈[3,20]**), și construiește în memorie un tablou bidimensional cu **n** linii și **n** coloane, având proprietățile:

- toate elementele situate pe diagonala secundară sunt nule;
- prima linie conține un șir strict descrescător de numere consecutive, iar ultima linie conține un șir strict crescător de numere consecutive;
- fiecare dintre celelalte linii conține, începând cu prima poziție, până la diagonala secundară inclusiv, de la stânga la dreapta, un șir strict descrescător de numere consecutive, iar începând de la diagonala secundară inclusiv, până la ultima poziție, de la stânga la dreapta, un șir strict crescător de numere consecutive.

Programul afișează pe ecran tabloul construit, fiecare linie a tabloului pe câte o linie a ecranului, cu elementele aflate pe aceeași linie separate prin câte un spațiu.
Exemplu: dacă **n=4** se afișează pe ecran tabloul alăturat. **(10p.)**

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

```
#include <iostream>
using namespace std;
//generare de matrice
int main()
{
   int a[21][21],n,i,j;
   cin>>n;
   int k;
   //pana la diagonala secundara pun k si imediat k--
   //dupa diagonala secundara pregatesc k++ si apoi pun k
   for(i=1; i<=n; i++)
   {
      k=n-i;//observ acest start pentru fiecare rand in parte
      for(j=1; j<=n; j++)
         if(i+j<n+1)//deasupra diagonalei secundare
         {
            a[i][j]=k;k--;
         }
         else if(i+j==n+1)
               {
                  a[i][j]=0;k=0;//restartez
               }
            else //au ramas cele de sub diagonala secundara
            {
               k++;a[i][j]=k;
            }
   }
   //afisare matrice
   for(i=1; i<=n; i++)
   {
      for(j=1; j<=n; j++)
         cout<<a[i][j]<<" ";
      cout<<endl;
   }
   return 0;
}
```

Fișierul **bac.in** conține un șir de cel puțin patru și cel mult 10^5 numere întregi nenule din intervalul $[-10^9, 10^9]$, dintre care trei sunt negative, iar restul pozitive. Numerele sunt separate prin câte un spațiu. O secvență este formată din termeni aflați pe poziții consecutive în șir, iar lungimea secvenței este egală cu numărul de termeni ai acesteia.
Se cere să se afișeze pe ecran lungimea unei secvențe din șirul aflat în fișier care conține o singură valoare negativă și un număr maxim de valori pozitive. Proiectați un algoritm eficient din punctul de vedere al memoriei utilizate și al timpului de executare.
Exemplu: dacă fișierul conține numerele **15 21 -61 9 870 -23 11 5 8 -81 5 14**
pe ecran se afișează **6** (corespunzător secvențelor **9 870 -23 11 5 8** sau **11 5 8 -81 5 14**).
a. Descrieți în limbaj natural algoritmul proiectat, justificând eficiența acestuia. **(2p.)**
b. Scrieți programul C/C++ corespunzător algoritmului proiectat. **(8p.)**

```
#include <iostream>
#include <fstream>//esential
using namespace std;
int main()
{
   ifstream f("bac.in");
   //voi parca in pn1 si pn2 cate 2 pozitii a doua valori negative
"alaturate, consecutive"
   int x,pn1,pn2,max;pn1=0; pn2=0; max=0;
   //e o problema de maxim
   int n=0;int pozitie=0;
   while(f>>x)
   {
      pozitie++;//fiecare x se afla pe pozitie
      if(x<0)
         {
           n++;//numar negativele
           if(n==1)//am gasit negativ
           {
              int numere=pozitie-(pn2-pn1)-1;//determin cate sunt, scad 1
ca sunt 2 negative acum
              if(numere>max)max=numere;//determin maximul
              pn1=pn2;//transfer de pozitii
              pn2=pozitie;
              n=0;//resetez n
           }
         }
   }
   cout<<max;
   f.close();
   return 0;
}
```