SUBIECTUL I **(20 de puncte)**

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

1. Expresia C/C++
`(x<17) || !(x<=18 || x>=20) || (x>21)`
are valoarea 0 pentru orice valoare a variabilei întregi `x` din mulțimea:

a. `{17,18,19}` **b.** `{17,18,20,21}` **c.** `{18,19,20}` **d.** `{18,19,20,21}`

2. Utilizând metoda backtracking sunt generate toate numerele din intervalul `[100,999]` cu proprietatea că au cifrele în ordine crescătoare și cifrele aflate pe poziții consecutive sunt de paritate diferită. Primele cinci soluții generate sunt, în această ordine, `123`, `125`, `127`, `129`, `145`. Indicați cel de al 9-lea număr generat.

a. `149` **b.** `167` **c.** `169` **d.** `189`

3. Fiecare dintre variabilele `A` și `B`, declarate alăturat, memorează coordonatele (`x` abscisa, iar `y` ordonata) câte unui punct în sistemul de coordonate `xOy`. Indicați o expresie C/C++ care are valoarea `1` dacă și numai dacă segmentul cu extremitățile în punctele corespunzătoare variabilelor `A` și `B` intersectează axa `Ox` a sistemului de coordonate.

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

a. `(A-y)*(B-y)<=0` **b.** `A.y*B.y<=0` **c.** `punct.y(A,B)<=0` **d.** `y.A*y.B<=0`

4. Un graf orientat cu `6` vârfuri, numerotate de la `1` la `6`, are arcele `(1,2)`, `(1,6)`, `(1,5)`, `(2,3)`, `(3,1)`, `(3,5)`, `(4,6)`, `(5,6)`, `(6,2)`. Indicați numărul de vârfuri care au gradul extern mai mare decât gradul intern.

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

5. Un graf neorientat are `50` de noduri și `32` de muchii. Indicați numărul maxim de componente conexe pe care le poate avea graful.

a. `25` **b.** `31` **c.** `33` **d.** `42`

SUBIECTUL al II-lea **(40 de puncte)**

1. **Algoritmul alăturat este reprezentat în pseudocod.**
S-a notat cu `a%b` restul împărțirii numărului natural `a` la numărul natural nenul `b` și cu `[c]` partea întreagă a numărului real `c`.

```
citește n,k
(numere naturale)
┌dacă k=0 atunci nr←-1
│altfel
│ nr←0
│ p←1
│┌repetă
│| c←n%10; n←[n/10]
│|┌dacă c%2=0 atunci
│||  nr←nr+c*p; p←p*10
│||altfel k←k-1
│|└■
│└până când n=0 sau k=0
└■
scrie nr
```

a. Scrieți numărul afișat în urma executării algoritmului dacă se citesc, în această ordine, numerele `812302105` și `4`. **(6p.)**

2020

b. Dacă pentru `k` se citește numărul `1`, scrieți trei numere din intervalul $[10^3, 10^4)$ care pot fi citite pentru `n`, astfel încât, pentru fiecare dintre acestea, în urma executării algoritmului, să se afișeze un număr format din două cifre identice. **(6p.)**

programul afiseaza numarul format din cifrele pare ala lui n in ordinea in care apar in n, in limita aparitiei a k numere impare in numar, incepand sa numar cifrele impare cu cifra unitatilor.

Exemplu de date de intrare pentru n care respecta conditia data: 2122, 3122, 1188

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

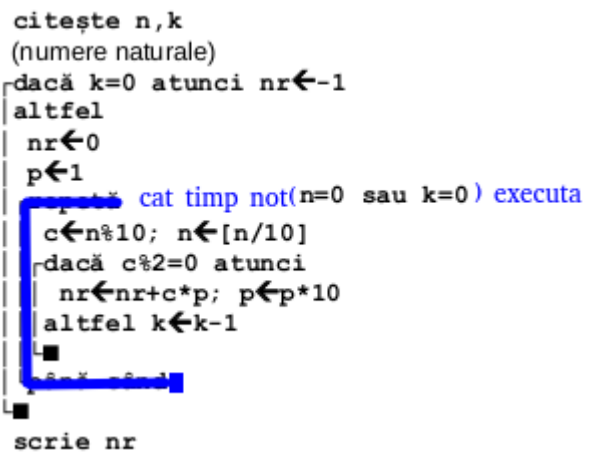
d. Scrieți în pseudocod un algoritm, echivalent cu cel dat, înlocuind adecvat structura **`repetă...până când`** cu o structură repetitivă de alt tip. **(6p.)**

c)

```
#include <iostream>
using namespace std;
unsigned long n,k,p,nr,c;

int main()
{
cin>>n>>k;
if(k==0) nr=-1;
  else
  {
   nr=0;p=1;
   do
   {
        c=n%10;n=n/10;
        if(c%2==0) {nr=nr+c*p;p=p*10;}
          else k=k-1;
   }while(!(n==0 || k==0));
  }
cout<<nr;
return 0;
}
```

d)



2. Subprogramul `f` este definit alăturat. Scrieți ce se afișează în urma apelului de mai jos. **(6p.)**

`f(5);`

5 3 1
1 2 3 4 5

```
void f(int n)
{ if (n!=0)
   { if (n%2==1) cout<<n<<' '; | printf("%d ",n);
     f(n-1);
     cout<<n<<' '; | printf("%d ",n);
   }
   else cout<<endl; | printf("\n");
}
```

3. Variabilele `s1` și `s2` pot memora câte un șir cu cel mult 20 de caractere. Scrieți ce se afișează în urma executării secvenței alăturate. **(6p.)**

```
strcpy(s1,"bacalaureat2020");
cout<<strlen(s1); | printf("%d",strlen(s1));
strcpy(s2,s1+11); strcpy(s1+3,s2);
cout<<s1; | printf("%s",s1);
```

```
strcpy(s1,"bacalaureat2020");
cout<<strlen(s1);
strcpy(s2,s1+11); strcpy(s1+3,s2);
cout<<s1;
```

Raspuns: 15bac2020

s1="bacalaureat2020"
scrie 15
s2="2020"; strcpy(s1+3,s2): bac 2020
=> s1="bac2020" s1 s2
scrie bac2020

SUBIECTUL al III-lea (30 de puncte)

1. 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 ($9000=2^3 \cdot 3^2 \cdot 5^3$). **(10p.)**

```
int nrDivPrimi(unsigned long n)
{
        unsigned long d;
        long c,nr=0;
```

```
        d=2;
        while(n>1)
        {
                c=0;
                while(n%d==0)
                {
                        n=n/d;
                        c=c+1;
                }
                if(c%2==1)nr++;
                d++;
        }
        return nr;
}
```

2. Scrieți un program C/C++ care citește de la tastatură două numere naturale din intervalul **[2,10²]**, **n** și **m**, și construiește în memorie un tablou bidimensional cu **n** linii și **m** coloane, cu proprietatea că parcurgându-l linie cu linie de sus în jos și fiecare linie de la stânga la dreapta, se obține șirul primelor **n*m** pătrate perfecte impare, ordonat strict descrescător, ca în exemplu.

Elementele tabloului obținut se afișează pe ecran, fiecare linie a tabloului pe câte o linie a ecranului, valorile de pe aceeași linie fiind separate prin câte un spațiu.
Exemplu: pentru **n=2**, **m=3** se obține tabloul alăturat. **(10p.)**

```
121  81  49
 25   9   1
```

```
#include<iostream>
using namespace std;
int a[100][100],n,m;
void init()
{
        for(int i=0;i<n;i++)
                for(int j=0;j<m;j++)
                        a[i][j]=0;
}

void tablou()
{
        int c=1;
        for(int i=n-1;i>=0;i--)
                for(int j=m-1;j>=0;j--)
                {
                        a[i][j]=c*c;c=c+2;
                }
}

void afis()
{
int i,j;
for(i=0;i<n;i++)
{
        for(j=0;j<m;j++)
                cout<<a[i][j]<<' ';
        cout<<endl;
}
}

int main()
{
        cin>>n>>m;
        init();
        tablou();
        afis();
        return 0;
}
```

3. Fișierul **bac.in** conține numere naturale: pe prima linie două numere din intervalul **[1,10⁶]**, **m** și **n**, pe a doua linie un șir de m numere din intervalul **[1,10⁹]**, iar pe a treia linie un șir de **n** numere din intervalul **[1,10⁹]**. Numerele aflate pe aceeași linie a fișierului sunt separate prin câte un spațiu, și ambele șiruri sunt ordonate crescător.
Se cere să se afișeze pe ecran, în ordine strict crescătoare, un șir format dintr-un număr maxim de termeni care aparțin cel puțin unuia dintre cele două șiruri, astfel încât oricare două elemente aflate pe poziții consecutive să fie de paritate diferită. Numerele afișate sunt separate prin câte un spațiu.

Proiectați un algoritm eficient din punctul de vedere al timpului de executare.
Exemplu: dacă fișierul are conținutul alăturat, se afișează pe ecran
2 3 4 5 8 11 14 sau **2 3 4 5 10 11 14**

```
8 5
2 4 5 8 8 11 14 14
3 4 5 5 10
```

a. Descrieți în limbaj natural algoritmul proiectat, justificând eficiența acestuia. **(2p.)**
b. Scrieți programul C/C++ corespunzător algoritmului proiectat. **(8p.)**

```
#include <iostream>
#include <fstream>
using namespace std;
ifstream f("bac.in");
long long n,m,k,i,t,j;
long long a[1000000],x;
int main()
{
k=1;
f>>m>>n;
f>>x;
a[k]=x;
for(i=2;i<=m;i++)
        {
                f>>x;
                if(a[k]!=x){k++;a[k]=x;}
        }
f>>x;
i=1;
long ultim,p;
if (a[1]<x) {i++; cout<<a[1]<<' ';ultim=a[1];}
else
        {cout<<x;ultim=x;}
while(i <= k && !f.eof())
        if(a[i] < x)
                if(ultim%2!=a[i]%2) { ultim=a[i];cout<<a[i]<<' ';i++; }
               else
                        i++;
                else
            if(ultim%2!=x%2){ultim=x; cout<<x<<' ';f>>x;}
       else f>>x;;

if(i <= k)
         for(p = i; p <=k; p++)
                if(ultim%2!=a[p]%2) {cout<<a[p]<<' '; ultim=a[p]; }
while(f>>x)
   if(ultim%2!=x%2) {cout<<x<<' '; ultim=x; }
f.close();
return 0;
}
```

Am folosit un vector pentru a retine valorile distincte din primul sir.
Datorita faptului ca elementele din cel de-al doilea sir sunt in ordine crescatoare vom folosi pentru citirea elementelor doar o singura variabila x care va retine valoarea elementului curent din al doilea sir.
Variabila ultim va retine valoarea ultimei valori afisate.

Vom lua pe rand, elemente din cele doua siruri, asemnator cu algoritmul folosit la interclasarea a doua siruri cu elementele in doua crescatoare.
Daca cea mai mica dintre cele doua valori are paritate diferita fata de ultima valoare afisata atunci se va afisa aceasta valoare si vom trece la urmatoarea valoare din sirul din care face parte.
Daca cea mai mica dintrele cele doua valori nu are o paritate diferita fata de ultima valoare afisata atunci se va trece la urmatoarea valoare din sirul din care face parte.
Cand unul dintre cele doua siruri se termina elementele ramase din celalalt sir vor fi afisate cu respectarea conditiei de paritate impuse.
De fiecare data se alege cate o valoare minima din cele doua siruri care au elementele in ordine crescatoare, deci, sirul rezultat, va avea elementele in ordine crescatoare.
Algoritmul nu are structuri repetitive imbricate deci timpul de executie este mic, complexitatea acestuia fiind O(max(n,m)).