SUBIECTUL I **(20 de puncte)**

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

1. Variabile **x**, **y** şi **z** sunt de tip întreg. Indicați o expresie C/C++ care are valoarea **1** dacă şi numai dacă **x**, **y** şi **z** au valori identice.

   a. `x==y && x==z` **b.** `x==y==z`

   c. `!( x!=y && x!=z)` **d.** `!(x!=y!=z)`

2. Subprogramul **f** este definit alăturat. Pentru apelul **f(20,2020)**, funcția se execută de:

```
int f(int x, int y)
{   if(x<=1 || y<=1) return 0;
    if(x>y) return 1+f(f(x/y,y),y);
    return 1+f(x,f(x,y/x));
}
```

   a. 5 ori **b.** 9 ori **c.** 11 ori **d.** 20 de ori

3. Utilizând metoda backtracking, se generează toate modalitățile de a pregăti clătite, folosind, într-o anumită ordine, toate ingredientele din mulțimea {**făină**, **lapte**, **ouă**} pentru aluat, apoi unul dintre ingredientele din mulțimea {**ciocolată**, **dulceață**, **urdă**} pentru umplutură, și, la final, unul dintre ingredientele din mulțimea {**cașcaval**, **mărar**, **frișcă**} pentru ornare, având în vedere următoarele restricții: frișca se poate folosi numai împreună cu ciocolata și dulceața, iar mărarul și cașcavalul numai împreună cu urda. Primele cinci soluții generate sunt, în această ordine: **(făină, lapte, ouă, ciocolată, frișcă)**, **(făină, lapte, ouă, dulceață, frișcă)**, **(făină, lapte, ouă, urdă, cașcaval)**, **(făină, lapte, ouă, urdă, mărar)**, **(făină, ouă, lapte, ciocolată, frișcă)**. Indicați a șaptea soluție generată.

   a. **(ouă, lapte, făină, urdă, mărar)** **b.** **(lapte, făină, ouă, ciocolată, frișcă)**

   c. **(făină, ouă, lapte, dulceață, frișcă)** **d.** **(făină, ouă, lapte, urdă, cașcaval)**

4. Numim înălțime a unui arbore cu rădăcină numărul de muchii traversate de cel mai lung lanț elementar cu una dintre extremități în rădăcina arborelui. Un arbore cu rădăcină, cu **8** noduri, numerotate de la **1** la **8**, este reprezentat prin vectorul "de tați" **(6,6,5,3,0,5,8,4)**. Indicați înălțimea arborelui:

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

5. Un graf neorientat are **10** noduri, numerotate de la **1** la **10**, și muchiile **[1,2]**, **[1,3]**, **[1,10]**, **[3,10]**, **[4,5]**, **[4,6]**, **[4,8]**, **[5,7]**, **[5,9]**, **[6,8]**, **[6,9]**, **[7,9]**, **[8,9]**. Indicați numărul minim de muchii care trebuie adăugate pentru ca graful obținut să aibă cel puțin un lanț eulerian (lanț care traversează toate muchiile grafului).

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

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**.

```
citeşte n (număr natural)
┌repetă
| c1←n%10; n←[n/10]; c2←n%10
| ┌dacă c1>c2 atunci
| | c2←c1; c1←n%10
| └■
| ┌cât timp c1<c2 execută
| | scrie c1
| | c2←[c2/2]
| └■
└până când n≤9
```

   a. Scrieţi ce se afișează în urma executării algoritmului dacă se citeşte numărul **49335**. **(6p.)**

   3334

   b. Scrieţi trei numere de patru cifre care pot fi citite astfel încât, pentru fiecare dintre acestea, în urma executării algoritmului, valoarea afişată să fie **1100**. **(6p.)**

   oricare 3 numere din {1014, 1015, 1016, 1017}

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

   d. Scrieţi în pseudocod un algoritm, echivalent cu cel dat, înlocuind adecvat structura **cât timp...execută** cu o structură repetitivă de alt tip. **(6p.)**

d)

```
citeşte n (număr natural)
┌repetă
| c1←n%10; n←[n/10]; c2←n%10
| ┌dacă c1>c2 atunci
| | c2←c1; c1←n%10
| └■
| repeta
| | scrie c1
| | c2←[c2/2]
| pana cand c1>=c2
| ■
└până când n≤9
```

c)

```
#include <iostream>
using namespace std;
int main(){
unsigned long n,c1,c2;
cin>>n;
do{
c1=n%10;n=n/10;c2=n%10;
if(c1>c2)
  {c2=c1;c1=n%10;}
while(c1<c2)
  {cout<<c1; c2=c2/2;}
}while(n>9);
return 0;}
```

2. Variabila **p** memorează date despre un poliedru regulat: numărul de vârfuri, lungimea muchiei și două unghiuri specifice (dintre o față și o muchie, respectiv dintre două fețe). Știind că expresiile C/C++ de mai jos au ca valori un număr natural reprezentând numărul de vârfuri ale poliedrului, și numerele reale reprezentând lungimea muchiei, respectiv cele două unghiuri specifice, scrieți definiția unei structuri cu eticheta **poliedru**, care permite memorarea datelor precizate, și declarați corespunzător variabila **p**.

p.NrVarfuri **p.Muchie** **p.Unghi.FataMuchie** **p.Unghi.FataFata** **(6p.)**

```
struct U{
    float FataMuchie;
    float FataFata;
}
```

```
struct poliedru {
    int NrVarfuri;
    float Muchie;
    struct U Unghi;
}p;
```

3. Variabilele **i** și **j** sunt de tip întreg, iar variabila **s** poate memora un șir de cel mult **20** de caractere. Scrieți șirul memorat de variabila **s** în urma executării secvenței de mai jos.

```
strcpy(s,"optsprezece"); i=0; j=strlen(s)-1;
while(i<j)
{ if(strchr("aeiou",s[i])==NULL && strchr("aeiou",s[j])!=NULL)
  {  s[i]=s[i]+1; s[j]=s[j]-1;}
  i=i+1;j=j-1;
}
```

opusqrdzdce **(6p.)**

SUBIECTUL al III-lea (30 de puncte)

1. 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∈[2,10⁹]**). 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 (unsigned long n)
{
        unsigned long a,b,gasit=0;
        for(a=2;a<=n;a++)
                for(b=1;b<=n;b++)
                        if(a*b+a/b==n)
                                {cout<<a<<'-'<<b<<' ';
                                gasit=1;}
if(!gasit) cout<<"nu exista";
}
```

2. Într-un tablou bidimensional, cu elemente având valori numai în mulțimea **{0,1}**, numim coloane „complementare" două coloane cu proprietatea că oricare două elemente ale acestora, aflate pe aceeași linie, sunt diferite.

Scrieți un program C/C++ care citește de la tastatură două numere naturale din intervalul **[2,20]**, **m** și **n**, și elementele unui tablou bidimensional cu **m** linii și **n** coloane, numere naturale din mulțimea **{0,1}**. Programul afișează pe ecran numărul de coloane ale tabloului care sunt „complementare" cu prima coloană a acestuia.

Exemplu: dacă **m=3**, **n=6**, pentru tabloul alăturat se afișează pe ecran **3**. **(10p.)**

1	1	0	0	1	0
0	1	1	1	1	1
1	0	0	0	1	0

```
#include <iostream>
using namespace std;
int a[21][21],m,n;
int main()
{
int i,j,c=0,da;
cin>>m>>n;
for(i=1;i<=m;i++)
        for(j=1;j<=n;j++)
```

```
        {
                cout<<"a["<<i<<','<<j<<"]=";
                cin>>a[i][j];
        }
for(j=2;j<=n;j++)
{
        da=1;
        i=1;
        while(i<=m && da==1)
        {
                if(a[i][j]==a[i][1]) da=0;
                i++;
        }
        if(da==1)c++;
}
cout<<c;
return 0;
}
```

3. Fişierul **bac.txt** conţine, **în ordine descrescătoare**, cel puţin două şi cel mult 10^6 numere naturale din intervalul $[0,10^9]$, separate prin câte un spaţiu. Se cere să se afişeze pe ecran, în ordine strict descrescătoare, separate prin câte un spaţiu, numai numerele care apar în fişier de exact două ori. Dacă nu există niciun astfel de număr, se afişează pe ecran mesajul **nu exista**. Proiectați un algoritm eficient din punctul de vedere al memoriei utilizate şi al timpului de executare.

Exemplu: dacă fişierul conţine numerele **100 50 50 50 49 49 36 16 16 12 10 10 9 7 7** pe ecran se afişează, în această ordine, numerele **49 16 10 7**

a. Scrieți programul C/C++ corespunzător algoritmului proiectat. **(8p.)**

b. Descrieți în limbaj natural algoritmul proiectat, justificând eficiența acestuia. **(2p.)**

```
#include <iostream>
#include <fstream>
using namespace std;
ifstream f("bac.txt");
int main()
{
unsigned long a,b,c;
int gasit=0;
f>>a; c=1;
while(f>>b)
{
        if(a==b)c++;
          else
            if(a!=b && c==2) {cout<<a<<' ';gasit=1;c=1;}
             else
               if(a!=b) c=1;
          a=b;
}
if(c==2){cout<<a;gasit=1;}
if(gasit==0) cout<<"nu exista";
f.close();
return 0;
}
```

Vom folosi patru variabile: a- retine valoarea precedenta citita din fisier, b – retine valoarea curenta citita din fisier, c- retine cate elemente egale sunt consecutive, gasit – este un “steag” care retine valoarea 0 daca nu s-a gasit niciun numar in conditiile date si gasit=1 daca s-a gasit cel putin un numar.

Datorita faptului ca numerele sunt in ordine descrescatoare in fisier iar noi le parcurgem secvential, in ordinea din fisier, atunci cand vom afisa valorile ele vor fii in ordine descrescatoare.

Cand vom gasi o valoare care sa se repete marim numarul de aparitii pentru valoarea respectiva (datorita faptului ca ea exista, numarul de aparitii pentru ea se intializeaza cu 1 (c=1)).

Daca avem doua valori consecutive egale (c==2) dar cea de-a treia nu este egala cu cele precedente inseamna ca trebuie sa afisam valoarea precedenta si sa reinitializam numarul de aparitii pentru noua variabila.

Programul contine putine variabile deci va fi nevoie de putin spatiu de memorie.

Programul contine doar o singura structura repetitiva pentru a citit si prelucra datele din fisier deci, va avea un timp de executie mic.