SUBIECTUL I **(20 de puncte)**

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

1. Variabila **x** este de tip întreg și memorează un număr nenul. Indicați expresia C/C++ cu valoarea **1** dacă **x** memorează un divizor al lui **2020**.

a. `2020/(2020/x)==0` **b.** `2020/(2020%x)==0`

c. `2020%(2020%x)==0` **d.** `2020%(2020/x)==0`

2. Variabila **a** memorează un tablou bidimensional cu **6** linii și **6** coloane, numerotate de la **0** la **5**, cu elemente numere întregi, iar toate celelalte variabile sunt întregi.

```
for(i=0;i<6;i++)
  for(j=0;j<6;j++)
   a[i][j]=(i%2)*j+(j%2)*i;
```

Indicați valoarea sumei elementelor de pe diagonala principală a tabloului construit în urma executării secvenței de mai sus.

a. 6 **b.** 12 **c.** 18 **d.** 30

3. Utilizând metoda backtracking se generează toate posibilitățile de a realiza o listă de **3** lucrări distincte ale lui George Enescu din mulțimea **{Oedip, Poema română, Rapsodia română nr. 1, Rapsodia română nr. 2, Simfonia nr. 1}**. Două liste sunt distincte dacă diferă prin cel puțin o lucrare sau prin ordinea acestora. Primele patru soluții generate sunt, în această ordine: **(Oedip, Poema română, Rapsodia română nr. 1)**, **(Oedip, Poema română, Rapsodia română nr. 2)**, **(Oedip, Poema română, Simfonia nr. 1)**, **(Oedip, Rapsodia română nr. 1, Poema română)**. Indicați penultima soluție generată.

a. **(Simfonia nr. 1, Rapsodia română nr. 2, Poema română)**

b. **(Simfonia nr. 1, Rapsodia română nr. 2, Oedip)**

c. **(Rapsodia română nr. 2, Rapsodia română nr. 1, Poema română)**

d. **(Rapsodia română nr. 2, Simfonia nr. 1, Oedip)**

4. Un arbore cu **10** noduri, numerotate de la **1** la **10**, este reprezentat prin vectorul de „tați" **(5,1,10,7,0,4,1,7,4,5)**. Indicați numărul total de descendenți ai nodului **7**.

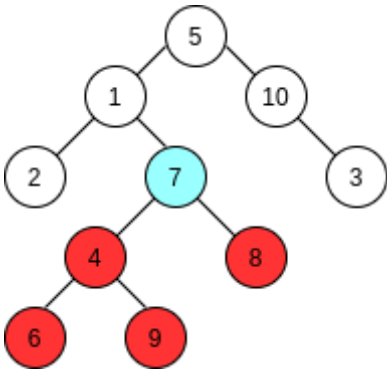
a. 1 **b.** 2 **c.** 3 **d.** 4

5. Un graf orientat cu **6** vârfuri, numerotate de la **1** la **6**, este reprezentat prin matricea de adiacență alăturată. Precizați numărul tuturor grafurilor parțiale distincte ale grafului dat. Două grafuri parțiale sunt distincte dacă matricele lor de adiacență sunt diferite.

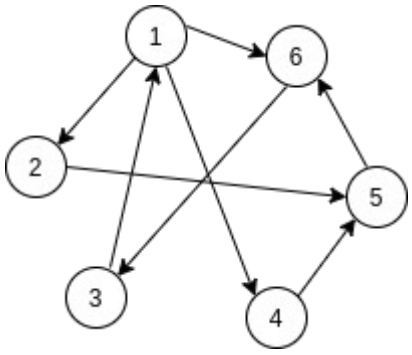
```
0 1 0 1 0 1
0 0 0 0 1 0
1 0 0 0 0 0
0 0 0 0 1 0
0 0 0 0 0 1
0 0 1 0 0 0
```

a. 2^6 **b.** 2^8 **c.** 2^{12} **d.** 2^{28}

4.



5.



Fie G un graf orientat sau neorientat cu n vârfuri şi m muchii/arce.

1. Numărul de grafuri parţiale ale lui G este 2^m.
2. Numărul de subgrafuri ale lui G este 2^n-1.
3. Numărul grafurilor orientate cu n vârfuri este $2^{n(n-1)} = 4^{n(n-1)/2}$
4. Numărul grafurilor orientate COMPLETE cu n vârfuri (exista cel putin un arc intre oricare 2 noduri) este $3^{n(n-1)/2}$.

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

1. **Algoritmul alăturat este reprezentat în pseudocod.**

```
citește n,k
  (numere naturale nenule)
pentru i←1,n execută
| pentru j←k,1,-1 execută
| | scrie j,' '
| 
| pentru j←1,k execută
| | scrie j,' '
| 
| k←k-1
```

a. Scrieți ce se afișează dacă se citesc, în această ordine, numerele 2 și 3. **3 2 1 1 2 3 2 1 1 2** **(6p.)**

b. Scrieți două seturi de valori din intervalul $[1,10^2]$ care pot fi citite astfel încât, pentru fiecare dintre acestea, în urma executării algoritmului, ultima valoare afișată să fie 20. **(6p.)**

(1,20) (2,21) (3,22) (4,23)... (11,30)...(31,50)... (47,66)...
. diferenta dintre k si n sa fie 19

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

d. Scrieți în pseudocod un algoritm, echivalent cu cel dat, înlocuind adecvat una dintre structurile **pentru...execută** cu o structură repetitivă de alt tip. **(6p.)**

c.

```
#include <iostream>
using namespace std;

int main()
{
unsigned int n,k,i,j;
cin>>n>>k;
for(i=1;i<=n;i++)
        {
        for(j=k;j>=1;j--)
                cout<<j<<' ';
        for(j=1;j<=k;j++)
                cout<<j<<' ';
        k--;
        }
return 0;}
```

d.

```
citește n,k
  (numere naturale nenule)
i=1
cat timp i<=n executa
| pentru j←k,1,-1 execută
| | scrie j,' '
| 
| pentru j←1,k execută
| | scrie j,' '
| 
| k←k-1
i=i+1
```

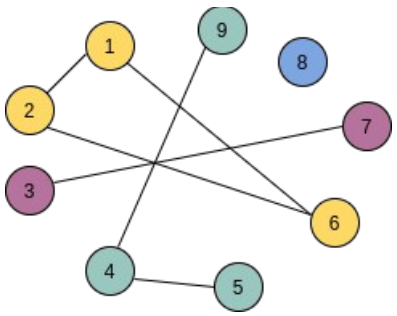
2. Subprogramul **f** este definit alăturat. Scrieți ce valori au **f(0)**, respectiv **f(23575209)**. **(6p.)**

0 3

```
int f (int n)
{ if (n!=0) return n%2+f(n/100);
  return 0;
}
```

3. Un graf neorientat cu **9** noduri, numerotate de la **1** la **9**, are muchiile **[1,2]**, **[1,6]**, **[2,6]**, **[3,7]**, **[4,5]**, **[4,9]**. Scrieți numărul componentelor conexe ale grafului și câte dintre acestea au un număr maxim de noduri. **2 componente: [1,2,6], [4,5,9]** **(6p.)**

4 componente conexe
2 componente cu numar maxim de noduri



SUBIECTUL al III-lea **(30 de puncte)**

1. Subprogramul **suma** are doi parametri:

- **n**, prin care primește un număr natural din intervalul $[0,10^9]$;
- **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**. **(10p.)**

```
void suma(unsigned long n, unsigned int &s)
{
        unsigned long nn;
        int i;
        s=0;
        for(i=0;i<=8;i+=2)
        {
                nn=n;
                while(nn!=0)
                {
                        if(i==nn%10){s=s+nn%10;nn=0;}
                        nn=nn/10;
                }
        }
}
```

2. Un cuvânt este **sufix** al unui alt cuvânt dacă se obține din acesta, prin eliminarea primelor sale litere.

Scrieți un program C/C++ care citește de la tastatură două numere naturale **n** și **k** (**n∈[2,20]**, **k∈[1,n]**) și apoi **n** cuvinte distincte, fiecare fiind format din cel mult **20** de caractere, numai litere mici ale alfabetului englez.

La introducerea datelor, după fiecare cuvânt se tastează Enter. Programul afișează pe ecran, separate prin câte un spațiu, cuvintele care îl au drept sufix pe al **k**-lea cuvânt citit, ca în exemplu. Dacă nu există astfel de cuvinte, se afișează pe ecran mesajul **nu exista**.

Exemplu: dacă **n=7**, **k=3** și se citesc cuvintele alăturate, pe ecran se afișează

paratirisi hiritisi **(10p.)**

```
isihast
paratirisi
isi
meremetisire
acolisitor
hiritisi
paraponisit
```

```
#include <string.h>
#include <iostream>
using namespace std;

int main()
{
  int n,k,t,tt,i,j,exista=0,p;
  char s[21][21],ss;
  cin>>n>>k;
  for(i=1;i<=n;i++)
                cin>>s[i];
  for(i=1;i<=n;i++)
  {
                tt=strlen(s[i]);t=strlen(s[k]);
                p=t-1;j=tt-1; //verificam literele cele doua cuvinte daca coincid, incepand de la sfarsitul sirurilor
                while(p>=0 && j>=0 && s[i][j]==s[k][p])
                        {p--;j--;}
                if(p==-1 && j>=0) //daca cuvantul s[k] est sufix al cuvantului s[i]
          {
                cout<<s[i]<<' ';
                exista=1;
          }
        }
  if(exista==0) cout<<"nu exista";
  return 0;
}
```

3. Numim **10-secvență** într-un șir de numere naturale, o succesiune de termeni aflați pe poziții consecutive în șir, cu proprietatea că sunt multipli ai numărului **10**. **Lungimea** secvenței este egală cu numărul de termeni ai săi.
Fișierul **bac.txt** conține un șir de cel mult 10^6 numere naturale din intervalul $[0,10^9]$, separate prin câte un spațiu. Cel puțin un termen din șir este multiplu al lui **10**. Se cere să se afișeze pe ecran două valori, separate printr-un spațiu, reprezentând lungimea maximă a unei **10**-secvențe din șirul aflat în fișier, respectiv numărul de 10-secvențe cu o astfel de lungime. Proiectați un algoritm eficient din punctul de vedere al memoriei utilizate și al timpului de executare.

Exemplu: dacă fișierul are conținutul alăturat, se afișează **3 2**	**7 3 200 100 10 9 6 100 1000 40 1002 20 30**

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

Variabilele folosite:
lmax = lungimea maxima a unei secvente
nrmax = numarul de secvente de lungime maxima
x = valoarea curenta citita din fisier
l = lungimea secventei curente
In problema se specifica faptul ca fisierul are cel putin un numar divizibil cu 10; ne vom pozitiona pe acel numar; in acest moment numarul de secvente cerut este 1, lungimea maxima ceruta este 1 si lungimea secventei curente este 1.
Vom citi pe rand numerele din fisier.
Daca numarul citit nu e divizibil cu 10 inseamna ca nu suntem intr-o secventa de numere divizibile cu 10 si lungimea secventei cerute este 0 pentru valoarea curenta citita din fisier.
Daca numarul curent citit din fisier este divizibil cu 10, marim numarul de numere divizibile cu 10, din secventa, cu 1.
Putem avea urmatoarele cazuri:
i. prin adaugarea acestui numar la secventa se ajunge la o secventa egala ca lungime cu secventa maxima caz in care marim cu 1 numarul de secvente de lungime maxima
ii. prin adaugarea acestui numar se obtine o secventa mai mare decat secventa maxima gasita caz in care reinitializam numarul de secvente ce respecta cerinta ceruta (nrmax=1) si lungimea secventei maxime cu lungimea noii secvente.
Problema foloseste un numar mic de variabile, nu foloseste vectori cu numar variabil de elemente prin urmare, memoria necesara pentru retinerea datelor la executarea programului este mica.
Programul nu foloseste structuri repetitive imbricate ci parcurge o singura data numerele din fisier, pentru a le citi si prelucra, pe rand. Prin urmare, timpul pentru executarea programului este mic.

```
#include <iostream>
#include <fstream>
using namespace std;

ifstream f("bac.txt");

int main()
{
        int lmax, l,nrmax;
        unsigned long x;
        f>>x;
        while(x%10!=0) f>>x;
        lmax=1;l=1;nrmax=1;
        while(f>>x)
        {
                if(x%10==0) l++;
                        else l=0;
                if(lmax==l) nrmax++;
                        else
                                if(l>lmax)
                                        {nrmax=1;lmax=l;}
        }
cout<<lmax<<' '<<nrmax;

f.close();
return 0;
}
```