Laborator 5
Grafuri neorientate

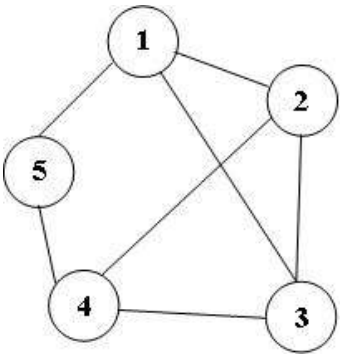
1. Graf partial

Fie G=(A,B) si $G_1=(A_1,B_1)$.

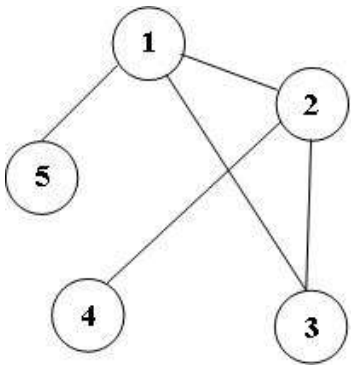
Spunem că G_1 este un graf parţial al lui G dacă $A=A_1$ şi B_1 este inclus sau egal cu B.

Un graf parţial se obţine dintr-un graf, îndepărtând o parte dintre muchiile sale şi păstrând toate nodurile acestuia.

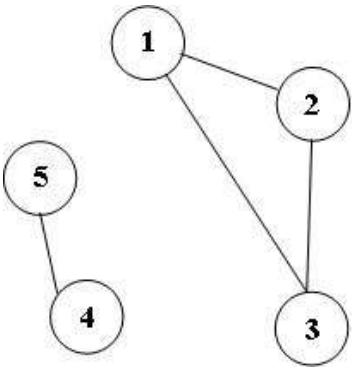
Exemplu 1:



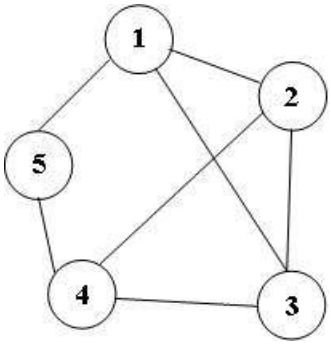
Graful iniţial



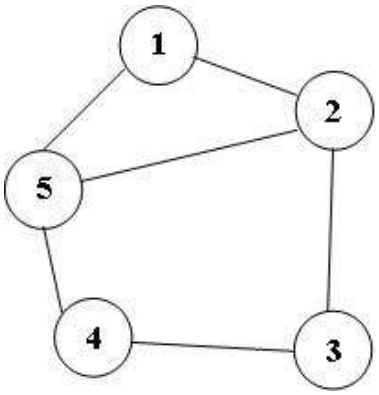
Da



Da



Da



Nu (are o muchie în plus)

2. Subgraful unui graf

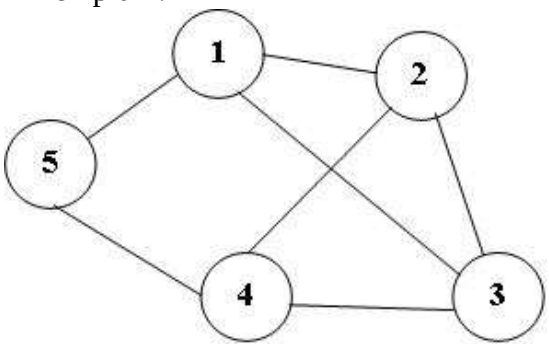
Fie G=(A,B) şi $G_1=(A_1,B_1)$;

A_1 inclus sau egal cu A; B_1 inclus sau egal cu B.

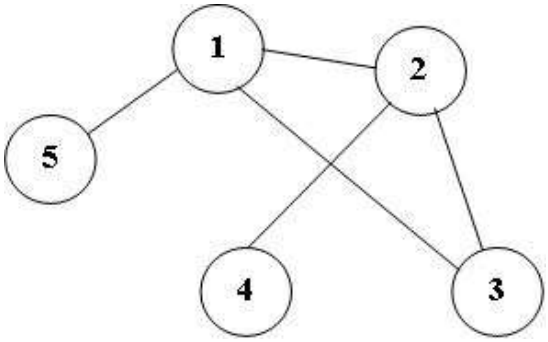
B_1 = {(x,y) / oricare x,y aparţine A_1 dacă (x,y) aparţine de B => (x,y) aparţine de B_1}

Subgraful se obţine din graful iniţial selectând o parte din nodurile sale şi o parte din nodurile adiacente cu acesta.

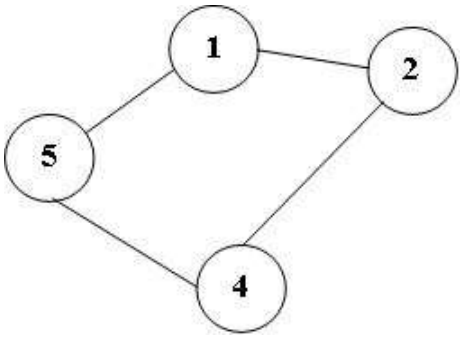
Exemplu 2:



Graful iniţial



Nu (nu are toate muchiile)

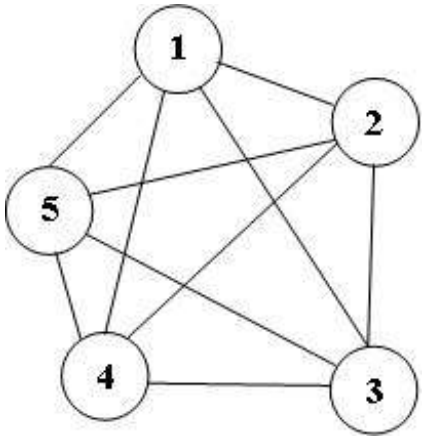


Da

3. Graf complet

Un graf este complet dacă oricare două vârfuri distince sunt adiacente.

Exemplu 3:



Un graf neorientat cu n noduri are n(n-1)/2 muchii.

Există un singur graf complet neorientat cu n noduri.

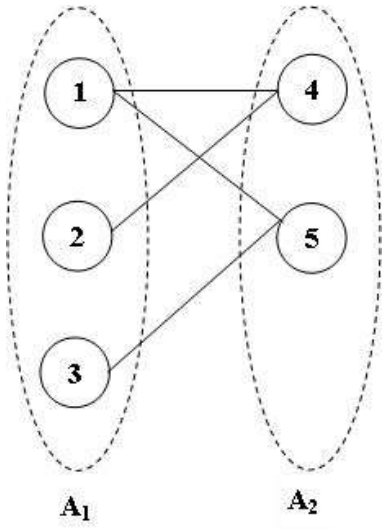
Există mai multe grafuri orientate complete cu n noduri.

4. Grafuri bipartite

Fie G=(A,B) neorientat.

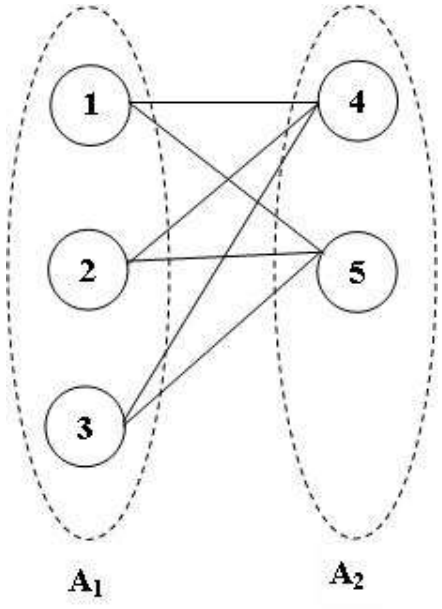
G este bipartit dacă există două mulţimi, A_1 şi A_2 astfel încât $A_1 \cap A_2 = Ø$ şi $A_1 \cup A_2 = A$, iar oricare muchie (x,y) aparţinând lui B are un capăt în mulţimea A_1 şi celălalt în A_2.

Exemplu 4:



Un graf bipartit este bipartit complet dacă fiecare nod din mulţimea A_1 este adiacent cu toate nodurile din A_2 şi reciproc.

Exemplu 5:

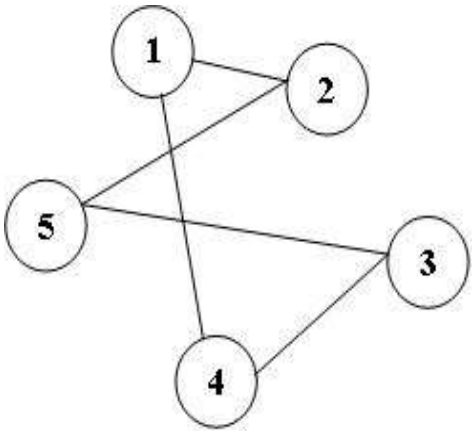


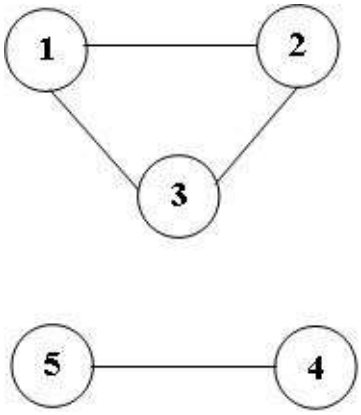
5. Grafuri conexe

Un graf este conex dacă este format dintr-un singur nod sau dacă între oricare două noduri ale sale există cel puţin un lanţ.

Exemplu 6:

Da





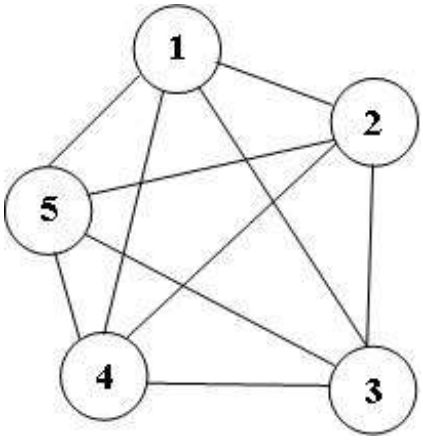
Probleme rezolvate

1) Sa se verifice daca un graf neorientat este complet.

```
#include<iostream.h>
int n,i,j,m,a[10][10];
int main()
{
        cout<<"Dati nr. de vf. ";cin>>n;
        cout<<"Dati elementele matricei de adiacenta"<<endl;
        for(i=1;i<=n-1;i++)
                for(j=i+1;j<=n;j++){
                        cout<<"a["<<i<<"]["<<j<<"]=";
                        cin>>a[i][j];
                        a[j][i]=a[i][j];
                }
        // determinarea nr. de muchii
        m=0;
        for(i=1;i<=n;i++)
                for(j=1;j<=n;j++)
                        if(a[i][j]==1) m++;
        m=m/2;
        if(m == n * (n-1)/2)            cout<<"Graf complet";
        else                    cout<<"Graful nu este complet";
}
```

Executia programului pentru urmatoarele date:

Dati nr. de vf. 5
Dati elementele matricei de adiacenta
a[1][1]=0
a[1][2]=1
a[1][3]=1
a[1][4]=1
a[1][5]=1
a[2][2]=0
a[2][3]=1
a[2][4]=1
a[2][5]=1
a[3][3]=0
a[3][4]=1
a[3][5]=1
a[4][4]=0
a[4][5]=1
Graf complet



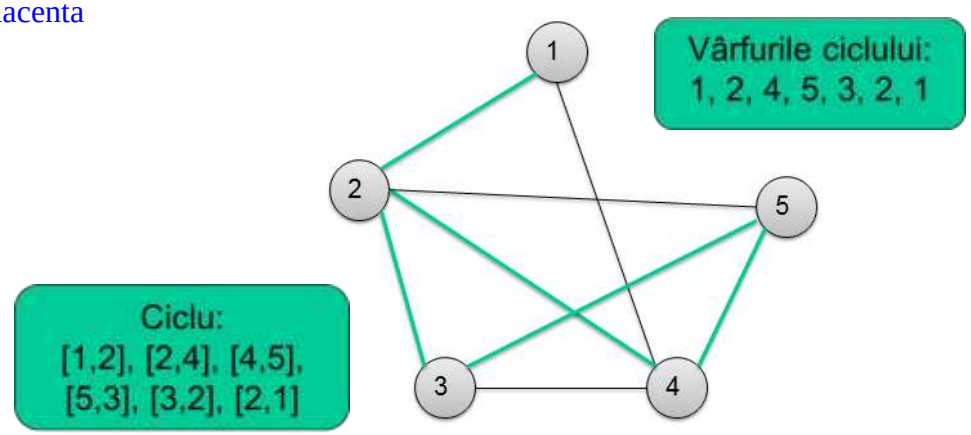
2) Sa se verifice daca o succesiune de varfuri date este ciclu pentru un graf. In caz afirmativ, sa se stabileasca daca este ciclu elementar sau nu.

```
#include<iostream.h>
int n,i,j,m,a[10][10],x[50];
int main()
{
        cout<<"Dati nr. de vf. ";cin>>n;
        cout<<"Dati elementele matricei de adiacenta"<<endl;
        for(i=1;i<=n-1;i++)
                for(j=i+1;j<=n;j++){
                        cout<<"a["<<i<<"]["<<j<<"]=";
                        cin>>a[i][j];
                        a[j][i]=a[i][j];
                }
        // citirea succesiunii de vf.
        cout<<"Dati nr. de muchii ";cin>>m;
        for(i=1;i<=m+1;i++)
        {
                cout<<"x["<<i<<"]=";
                cin>>x[i];
        }
        // verificare daca este lant: vf. consecutive sa fie muchii in matrice
        int lant=1;
        for(i=1;i<=n-1;i++)
                        if(a[x[i]][x[i+1]]==0) lant=0;
        // primul si ultimul vf. sunt egale
        int ok=1;
        if(lant==1)
                if(x[1]!=x[m]) ok=0;

        // muchii distincte
        int ciclu=1;
        if(ok==1)
                for(i=1;i<=m-2;i++)
                        for(j=i+1;j<=m;j++)
                          if(x[i]==x[j]  &&  x[i+1]==x[j+1]  ||  x[i]==x[j+1]  &&  x[i+1]==x[j])
ciclu=0;
        // verificare daca este ciclu elementar: vf. sa fie distincte doua cate doua, mai putin
primul si ultimul
        if(ciclu==1 && x[1]==x[m])
                cout<<"Ciclu elementar";
        else
                cout<<"Ciclu neelementar";
}
```

Executia programului pentru urmatoarele date:

Dati nr. de vf. 5
Dati elementele matricei de adiacenta
a[1][1]=0
a[1][2]=1
a[1][3]=0
a[1][4]=1
a[1][5]=0
a[2][2]=0
a[2][3]=1
a[2][4]=1
a[2][5]=1
a[3][3]=0
a[3][4]=1
a[3][5]=1
a[4][4]=0
a[4][5]=1
Dati nr. de muchii 6
x[1]=1
x[2]=2
x[3]=4
x[4]=5
x[5]=3
x[6]=2
x[7]=1
Ciclu neelementar



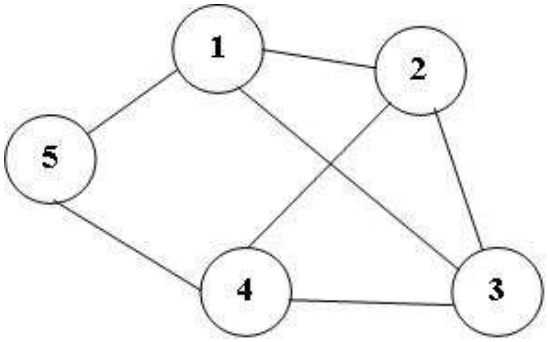
3) Se citesc 2 grafuri neorientate, unul cu n noduri si m muchii, iar celalalt cu k varfuri si l muchii, ambele date prin vectorul muchiilor. Sa se determine daca al doilea graf este subgraf al primului.

```
#include<fstream.h>
fstream f("date1.in",ios::in);
fstream g("date1.in",ios::in);

int a[100][100], b[100][100], n,m,k,l;

int subgraf()
 {
  for(int i=1;i<=k;i++)
    for(int j=1;j<=k;j++)
       if(a[i][j]!=b[i][j]) return 0;
 return 1;
 }
```

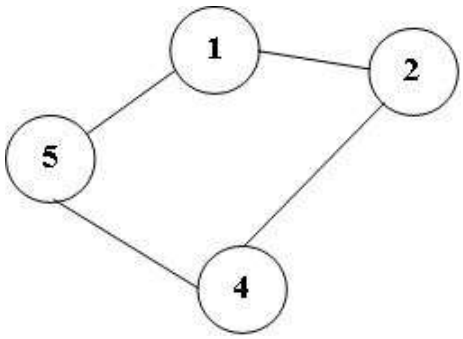
Date2.in
5 7
1 2
1 3
1 5
2 3
2 4
3 4
4 5
4 4
1 2
1 5
2 4
4 5



```
int main(void)
{
        int x,y,i;
        f>>n>>m;
        for(i=1;i<=m;i++)
        {
                f>>x>>y;
                a[x][y]=1;
                a[y][z]=1;
        }
        g>>k>>l;
        for(i=1;i<=l;i++)
        {
                g>>x>>y;
                b[x][y]=1;
                b[y][x]=1;
        }

        if(subgraf()) cout<<"da";
        else cout<<"nu";
f.close();
g.close();
}
```

Date2.out
da



4) Memorarea grafurilor neorientate folosind cele trei modalitati de reprezentare:
   a) Matrice de adiacenta
   b) Vector de muchii
   c) Liste de vecini

```
#include <fstream.h>
#include <vector.h>
ifstream fin("date.in");
ofstream fout("date.out");
struct muchie
{
   int i,j;
};
int A[50][50];          //      matrice de adiacenta
muchie M[1000];         //      vector muchiilor
int V1[50], V2[50];     //      liste de vecini
int n,m;                //n – nr. de noduri, m – nr. de muchii

void citire()
{
  int x,y,i;
  fin>>n>>m;
```

```
  for(i=1;i<=m;i++)
  {
     fin>>x>>y;
     M[i].i=x; M[i].j=y;
     A[x][y]=A[y][x]=1;
     V1[i]=x;
     V2[i]=y;
  }
}

void afisare()
{
   int i,j;
   fout<<"matricea de adiacenta:\n";
   for(i=1;i<=n;i++)
   {
      for(j=1;j<=n;j++)
      fout<<A[i][j]<<" ";
      fout<<endl;
   }
   fout<<"lista muchiilor:\n";
   for(i=1;i<=m;i++) fout<<M[i].i<<" "<<M[i].j<<endl;
   fout<<"lista vecinilor:\n";
   for(i=1;i<=n;i++)
   {
      fout<<i<<": ";
      fout<<V1[i]<<","<<V2[i]<< " ";
      fout<<endl;
   }
   fin.close();
   fout.close();
}

int main()
{
   citire();
   afisare();
}
```

Laborator 5

5) Se citeste dintr-un fisier de pe primul rand o valoare n reprezentand numarul de noduri pentru un graf neorientat si de pe urmatoarele n linii si coloane matricea de adiacenta corespunzatoare grafului.

a) Identificati multimea X
b) Identificati multimea U
c) Calculati gradele nodurilor impare
d) Verificati daca graful are varfuri izolate, daca da afisati nodul, daca nu afisati un mesaj

```
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
int a[20][20];
int main()
{
        ifstream f("matricegraf.in");
        int n,i,j,k;
        f>>n;
        //a)
        for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
        f>>a[i][j];
        cout<<"X=";
        for(i=1;i<=n;i++)
        cout<<i<<" ";
        cout<<endl;
        //b)
        cout<<"U=";
        for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
        if(a[i][j]==1&&i<j) cout<<"("<<i<<","<<j<<")";
        //c)
        for(i=1;i<=n;i++)
        {
                k=0;
                for(j=1;j<=n;j++)
                  if(i%2==1&&a[i][j]==1) k++;
                 if(i%2==1) cout<<endl<<"gradul nodului "<<i<<" este "<<k<<endl;
        }
        //d)
        for(i=1;i<=n;i++)
        {
          r=0;
          for(j=1;j<=n;j++)
            if(a[i][j]==1)       return 0;
}
```

Laborator 5

6) Se citeste din fisierul graf.in de pe prima linie nr de varfuri si numarul de muchii(n,m) de pe urmatoarele m randuri perechi de varfuri reprezentand muchiile grafului.
Se cere:
a) Sa se construiasca matricea de adiacenta si sa se scrie aceasta in fisierul MAT.OUT
b) calculati gradul fiecarui nod si pastrati acest grad intr-un vector
c) verificati daca graful are varfuri izolate

```
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
int a[20][20],v[40];
int main()
{
        ofstream g("mat.out");
        ifstream f("graf.in");
        int j,i,n,m,h,r=0,x,y,ok=0;
        f>>n;
        f>>m;
        // a)
        for(i=1;i<=m;i++)
        {
          f>>x;
          f>>y;
          a[x][y]=a[y][x]=1;
        }
        for(i=1;i<=n;i++)
        {
           for(j=1;j<=n;j++)
              g<<a[i][j]<<" ";
              g<<endl;
        }
        //b)
        for(i=1;i<=n;i++){
          h=0;
           for(j=1;j<=n;j++)
              if(a[i][j]==1) h++;
            r++;
            v[r]=h;
        }
        for(i=1;i<=n;i++)
        cout<<"nodul "<<i<<"are rangul "<<v[i]<<endl;
        // c)
        for(i=1;i<=n;i++)
           if(v[i]==0) ok=0;
           else ok=1;
         if(ok==0) cout<<"Graful are varfuri izolate"<<" ";
        else cout<<"Graful nu are varfuri izolate"<<" ";
}
```

7) Din fisierul GRAF.TXT, cititi de pe primul rand nr de noduri si nr de muchii, si de pe urmatoarele m randuri perechiile de noduri reprezentand muchiile. Formati matricea de adiacenta. enumerati nodurile terminale,
nodurile intermediare

```
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
int a[20][20];
int main()
{
        int i,j,n,m,x,y,k;
        ifstream f("graf.txt");
        f>>n;
        f>>m;
        for(i=1;i<=n;i++)
          for(j=1;j<=n;j++)
         {
            f>>x; f>>y;
            a[x][y]=1;
            a[y][x]=1;}
           for(i=1;i<=n;i++)
          {
            for(j=1;j<=n;j++)
                  cout<<a[i][j]<<" ";
              cout<<endl;
        }
        for(i=1;i<=n;i++)
        {
           k=0;
          for(j=1;j<=n;j++)
              if(a[i][j]==1) k++;
           if(k==1) cout<<"Nod terminal ="<<i<<endl;
           if(k>1) cout<<"Nod intermediar= "<<i<<endl;
        }
}
```

Probleme propuse spre rezolvare

1. Să se ruleze programele prezentate mai sus, urmărind apelurile şi valorile parametrilor de apel.

2. Se citeste un graf de la tastatura: numarul de noduri, numarul de muchii si muchiile.
a) sa se afiseze matricea de adiacente
b) Sa se determine gradul unui nod citit
c) Sa se afiseze pentru un nod citit nodurile adiacente
d) sa se afiseze nodurile incidente cu cea de a x muchie din matrice
e) sa se afiseze pentru fiecare nod gradul
f) sa se afiseze nodul (nodurile) avand cei mai multi vecini
g) sa se afiseze nodurile izolate

3. Sa se determine daca o matrice citita de la tastatura poate fi matricea unui graf neorientat. In caz afirmativ se va determina cate muchii are graful.

4. Sa se faca un program de numarare a gradelor fiecarui vârf dintr-un graf. Sa se determine apoi:
a) daca exista vârfuri izolate si vârfuri terminale;
b) toate vârfurile de grad maxim.

5. Sa se determine toate grafurile partiale ale unui graf.

6. Doua grafuri G=(X,U) si H=(Y,V) se numesc izomorfe daca exista o bijectie $f: X \rightarrow Y$ astfel încât $[x,y] \in U$ daca si numai daca $[f(x),f(y)] \in V$. Deci doua grafuri izomorfe au acelasi numar de vârfuri si se obtin unul din celalalt printr-o renumerotare a vârfurilor. Sa se faca un program care sa verifice daca doua grafuri sunt izomorfe.

7. Fiind dat un graf G = (X, U), sa se determine un lant simplu de lungime maxima.

8. Fiind dat un graf G = (X, U), sa se faca un program care sa verifice daca graful este bipartit complet.

9. Sa se faca un program de generare a tuturor ciclurilor unui graf. Observatie: puteti adapta programul realizat pentru generarea tuturor ciclurilor elementare.

10. Fie G un graf neorientat, cu n vârfuri şi m muchii, reprezentat prin matricea de adiacenţă. Să se realizeze programe, în C/C++, care:
   a) afişează gradele tuturor vârfurilor;
   b) afişează vârfurile de grad par;
   c) afişează vârfurile izolate;
   d) afişează vârfurile terminale;
   e) verifică dacă graful are vârfuri izolate;

t) verifică dacă graful are vârfuri terminale;
g) verifică dacă graful are vârfuri interioare (nu sunt nici izolate nici terminale);
h) verifică dacă graful are toate vârfurile izolate;
i) verifică dacă graful are toate vârfurile interioare (nu sunt nici izolate nici terminale):
j) afişează gradul unui vârf dat:
k) afişează vecinii unui nod dat, vf;
l) verifică dacă un vârf dat este terminal, izolat sau interior;
m) afişează gradul cel mai mare şi toate vârfurile care au acest grad
n) afişează frecventa vârfurilor:

- ✓ izolate: n1
- ✓ terminale: n2
- ✓ interioare: n3

o) fiind dat şirul g1, ...,gn, să se verifice dacă poate reprezenta şirul gradelor vârfurilor în această ordine;
p) fiind dat şirul g1 ...,gn, să se verifice dacă poate reprezenta şirul gradelor vârfurilor (nu neapărat în această ordine).