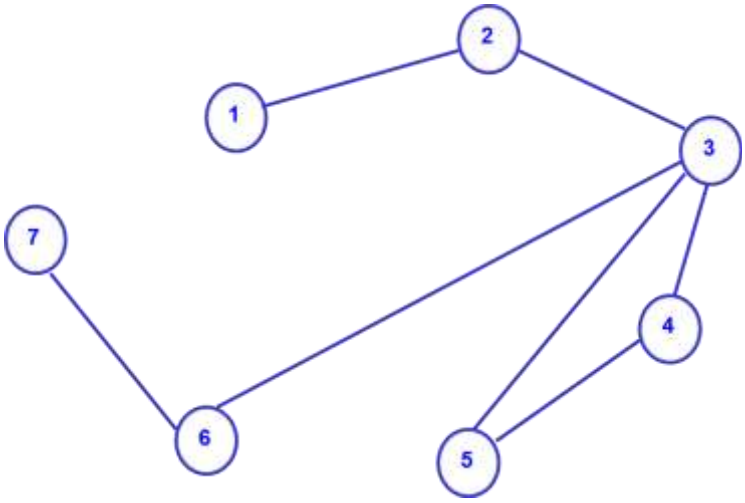
Laborator 5
Grafuri neorientate. Parcurgeri in grafuri neorientate:
1. Parcurgerea in latime (*Breadth First*)
2. Parcurgerea in adancime (*Depth First*)

Probleme rezolvate

1. Algoritmul de parcurgere in latime (Breadth First):

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

int viz[30],n,i,j,k,u,v,p,a[20][20],c[30];
int main(void)
{
cout<<"Dati numarul de varfuri n = ";
cin>>n;cout<<n<<'\n';
for(i=1; i<=n-1; i++)
  for(j=i+1; j<=n; j++){
        cin>>a[i][j];
        a[j][i] = a[i][j];
  }
        cout<<"Dati varful de plecare ";    cin>>i;cout<<i<<'\n';
  for(j=1; j<=n; j++) viz[j]=0;
  c[1]=i;
  p=1;
  u=1;
  viz[i]=1;
  while(p<=u){
        v=c[p];
        for(k=1; k<=n; k++){
              if( (a[v][k]==1) && (viz[k]==0) ){
                    u++;
                    c[u]=k;
                    viz[k]=1;
              }
        }
        p++;
  }
        cout<<"Lista varfurilor in parcugerea in latime: \n";
cout<<i<<" ";
for(j=2; j<=u; j++) cout<<c[j]<<" ";
        return 0;
}
```

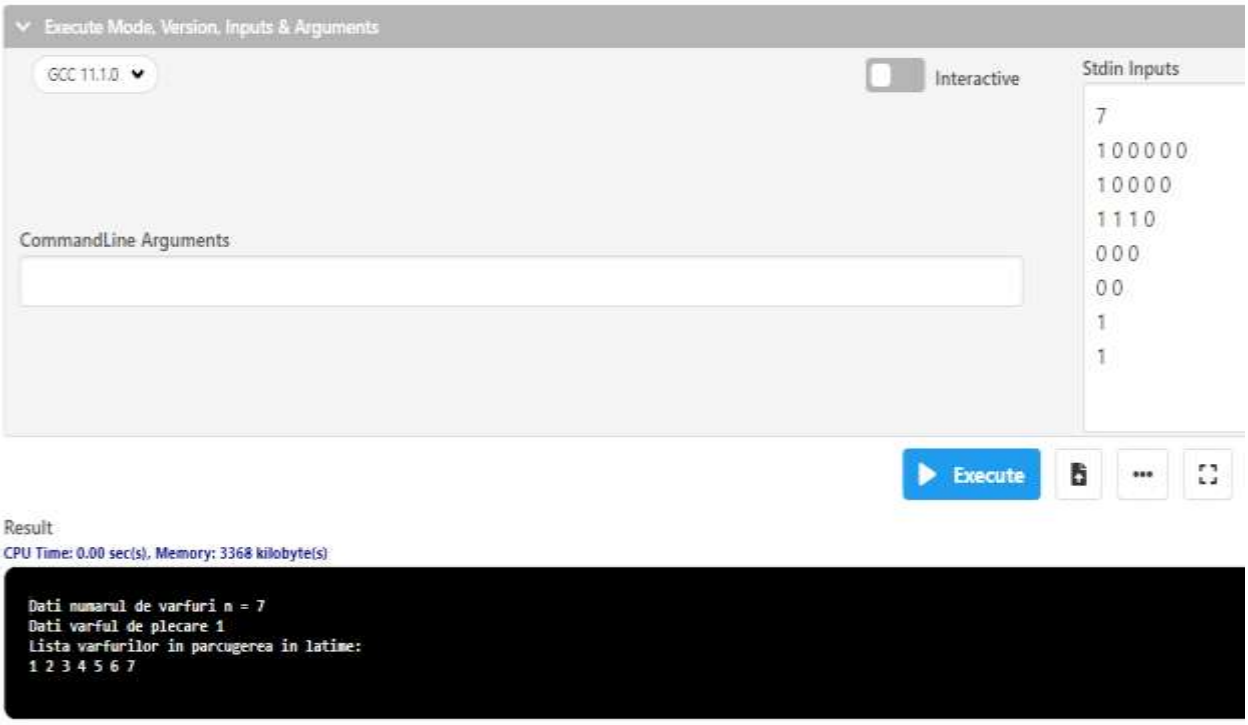
Sa se aplice acest algoritm pentru graful G={X,U}, unde X={1,2,3,4,5,6,7} si U={(1,2), (2,3), (3,4), (3,5), (3,6), (6,7)}



Varful de plecare este 1

Parcurgerea in latime permite afișarea următoarelor varfuri: 1, 2, 3,4, 5, 6, 7

Executia programului folosind compilatorul online https://www.jdoodle.com/online-compiler-c++/:



2. Parcurgerea in latime – varianta recursiva:

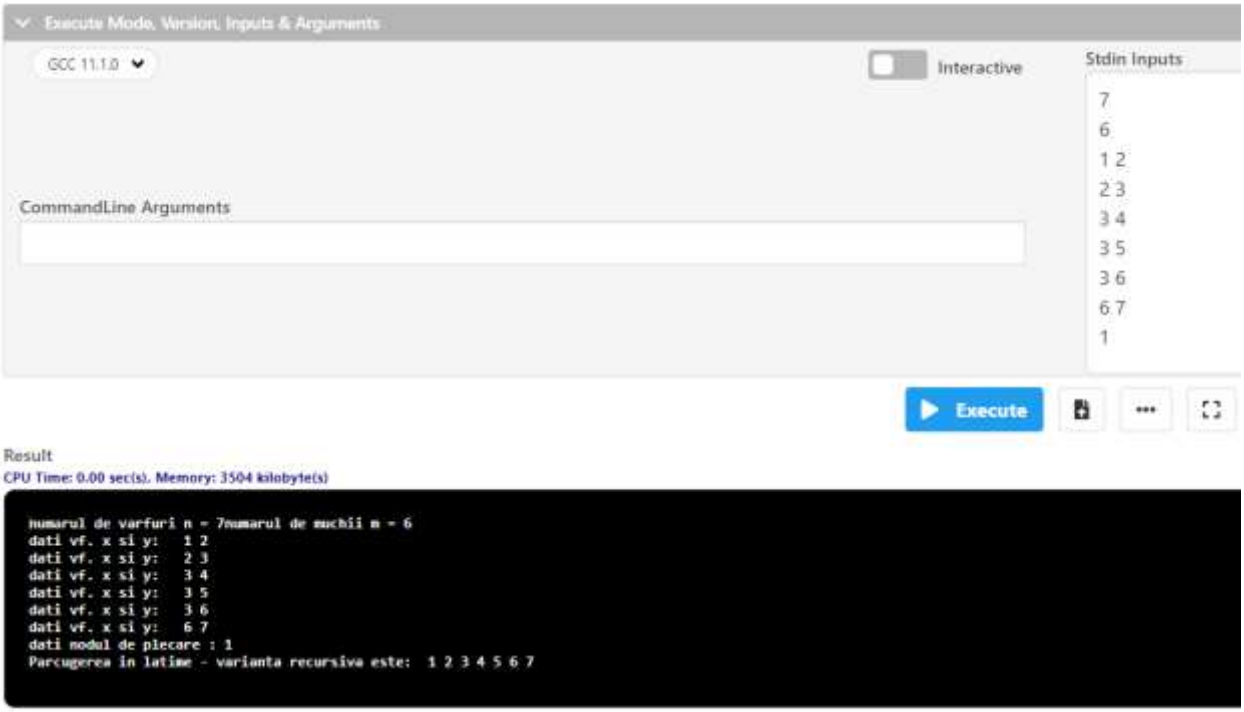
Se construiește o funcție numită BF_recursiva cu un parametru formal - i, care reprezintă poziția curentă la care s-a ajuns în coadă
Algoritmul este următorul:
se parcurg nodurile grafului, cu j:

- dacă j este adiacent cu nodul curent din coadă și j este nevizitat
- atunci se adaugă la coadă;
- și apoi se marchează ca fiind vizitat;
- dacă mai sunt elemente în coadă se trece la următorul și se reapelează funcția

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

int a[20][20];
int coada[20], viz[20];
int i, n, j, u, nod_plecare, m, x, y;

void BF_recursiva(int i)
{
int j,v;
for (j=1;j<=n;j++)
   {
      v=coada[i];
         if ((a[v][j]==1) && (viz[j]==0))
         {
                  u=u+1;
                  coada[u]=j;
                  viz[j]=1;
         }
   }
if (i<=u) BF_recursiva(i+1);
}
int main()
{
cout<<"numarul de varfuri n = ";
cin>>n;cout<<n;
cout<<"numarul de muchii m = ";
cin>>m;cout<<m;
for (i=1;i<=m;i++)
{
         cout<<"\ndati vf. x si y:   "; cin>>x>>y;cout<<x<<" "<<y;
         a[x][y]=1; a[y][x]=1;
}
for (i=1;i<=n;i++) viz[i]=0;
cout<<"\ndati nodul de plecare : ";
cin>>nod_plecare;cout<<nod_plecare<<'\n';
```

```
viz[nod_plecare]=1;
coada[1]= nod_plecare;
u = 1;
BF_recursiva(1);
cout<<"Parcugerea in latime - varianta recursiva este:  ";
for(i=1; i<=u; i++) cout<<coada[i]<<" ";
   return 0;
}
```

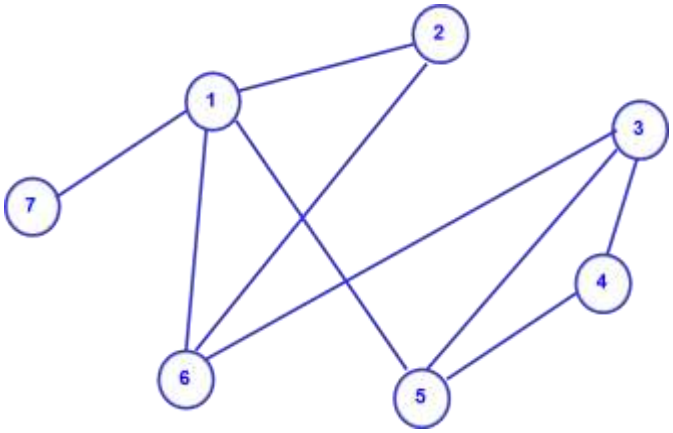
Executia programului folosind compilatorul online https://www.jdoodle.com/online-compiler-c++/:



3. Algoritmul de parcurgere in latime (Depth First):
Codul sursa al algoritmului de parcurgere in adancime:

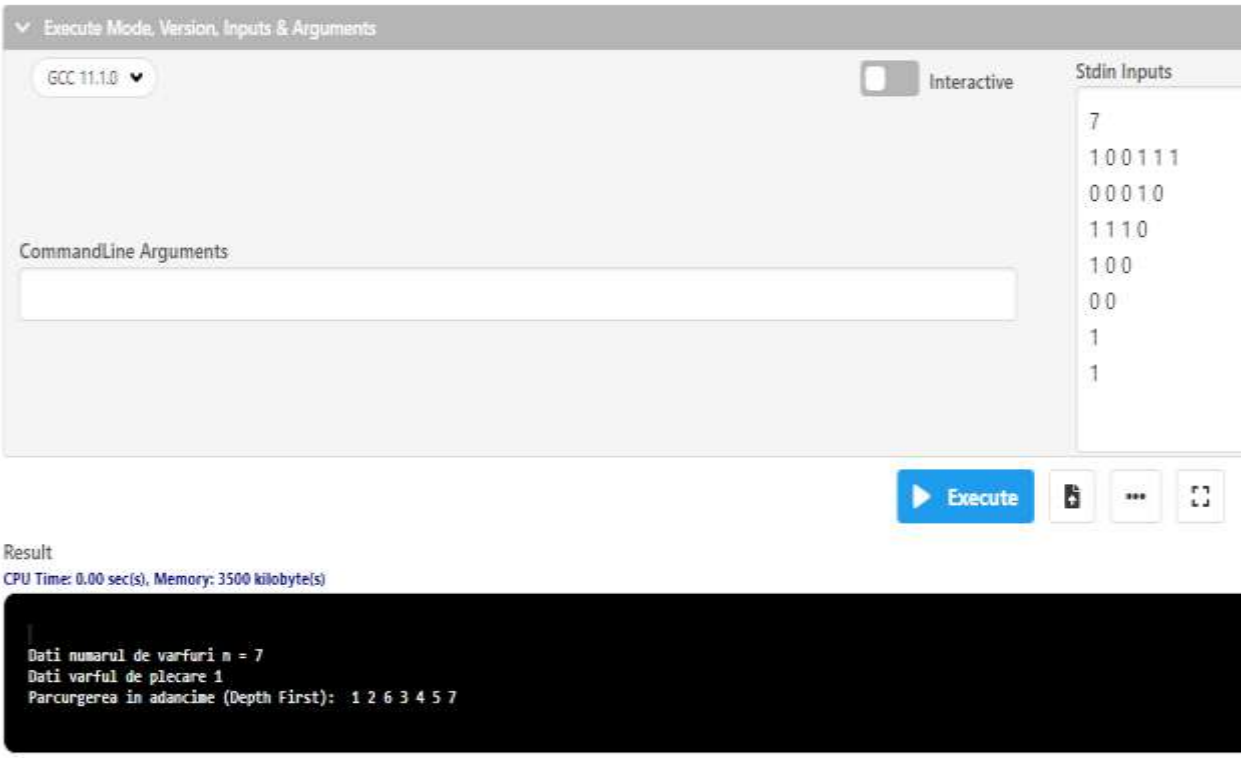
```
#include<iostream>
using namespace std;

int n, m, i, j, p, a[20][20], viz[30];
void df(int k)
{
int j;
cout<<k<<" ";
viz[k]=1;
for(j=1; j<=n; j++)
  if( (a[k][j]==1) && (viz[j]==0) ) df(j);
return;
}
int main(void)
{
cout<<"\nDati numarul de varfuri n = ";
cin>>n;cout<<n;
for(i=1; i<=n-1; i++)
  for(j=i+1; j<=n; j++)
  {
          cin>>a[i][j];
          a[j][i]=a[i][j];
  }
 cout<<"\nDati varful de plecare ";
 cin>>p;cout<<p;
 cout<<"\nParcurgerea in adancime (Depth First):  ";
 df(p);
    return 0;
}
```

Să considerăm următorul graf neorientat:



Metoda *DF*, aplicată acestui graf, pornind de la *vârful inițial 1*, conduce la vizitarea varfurilor în următoarea ordine:
1,2,6,3,4,5,7

Executia programului folosind compilatorul online https://www.jdoodle.com/online-compiler-c++/:

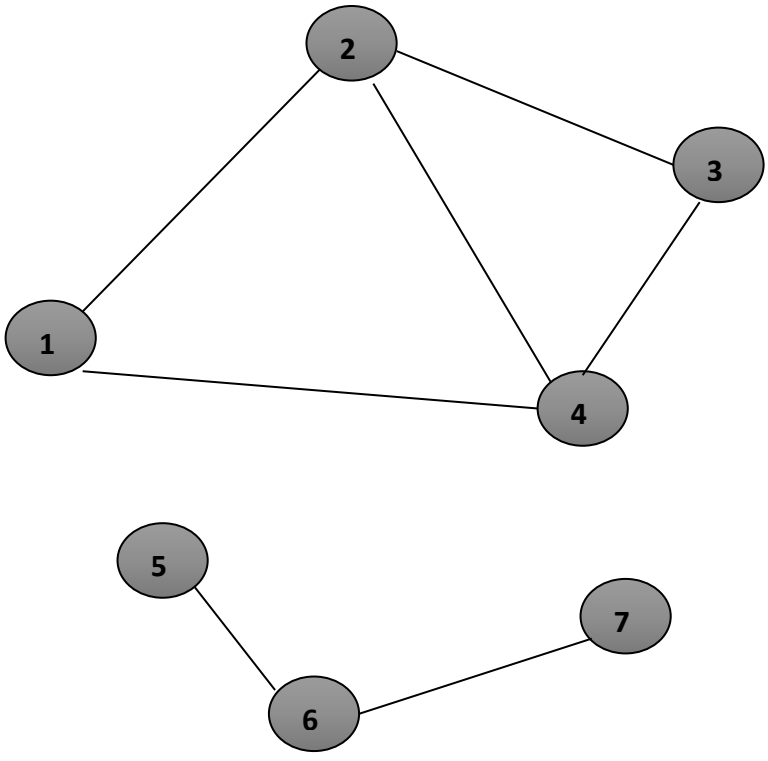


4. Se da un graf G={X,U} cu n varfuri si m muchii. Sa de determine componentele conexe ale grafului G.

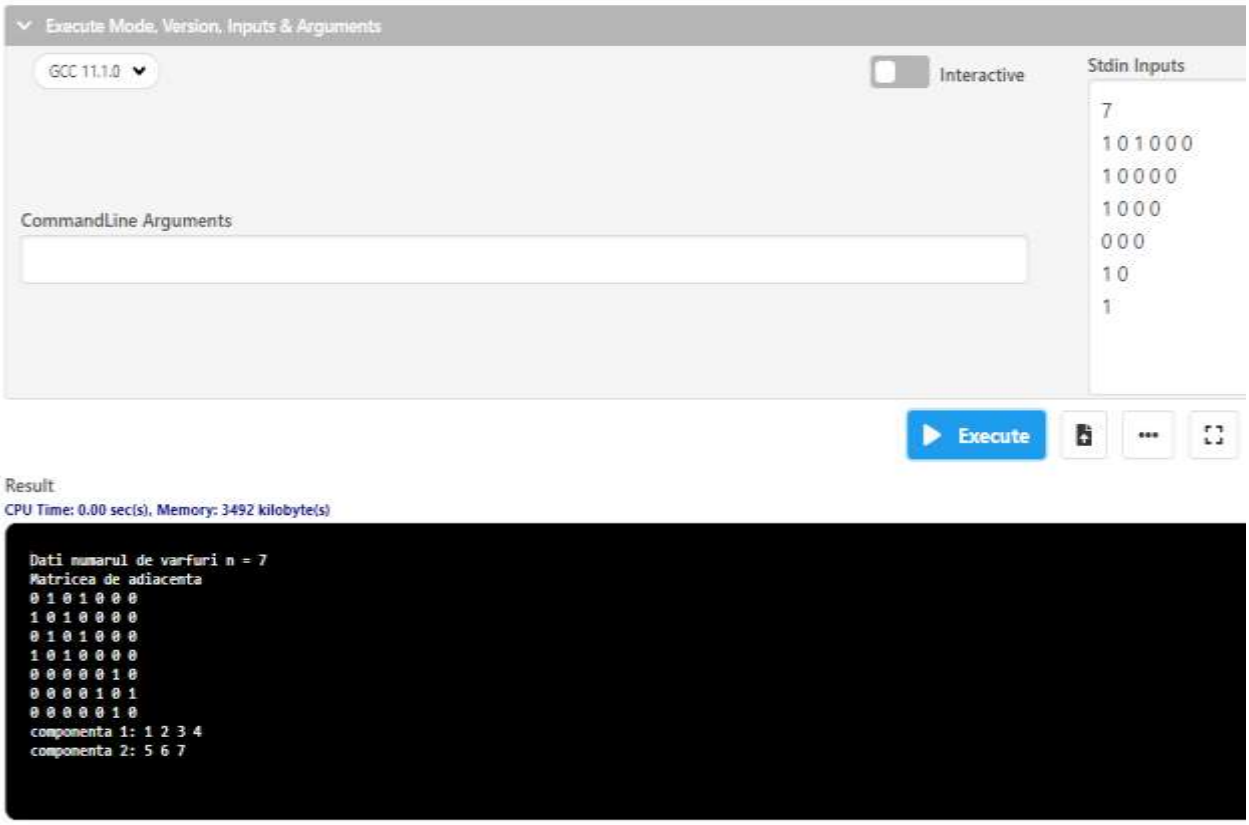
```
#include <iostream>
using namespace std;

int a[20][20],cc[30];
int n,ncc,i,j,v;          // ncc=nr. de componente conexe
void df(int v)
{
  int i;
  cc[v]=ncc;
  for(i=1;i<=n;i++)
          if( (a[v][i]==1) && (cc[i]==0) )
                        df(i);
}

int main(void)
{
cout<<"Dati numarul de varfuri n = ";
```

```
cin>>n; cout<<n;
cout<<"\nMatricea de adiacenta "<<endl;
for(i=1;i<=n-1;i++)
  for(j=i+1;j<=n;j++)
  {
        cin>>a[i][j];
        a[j][i]=a[i][j];
  }
for(i=1;i<=n;i++){
   for(j=1;j<=n;j++)    cout<<a[i][j]<<" ";
   cout<<'\n';
}
        ncc = 0;
for(i=1;i<=n;i++) cc[i]=0;
for(i=1;i<=n;i++)
        if (cc[i]==0) {
                ncc=ncc+1;
                df(i);
        }
for(i=1;i<=ncc;i++){                    // ncc = nr. de comp. conexe
        cout<<"componenta "<<i<<": ";
        for(j=1;j<=n;j++)
                if(cc[j]==i)    cout<<j<<" ";
        cout<<endl;
}
   return 0;
}
```

Să considerăm următorul graf neorientat:



Executia programului folosind compilatorul online https://www.jdoodle.com/online-compiler-c++/:

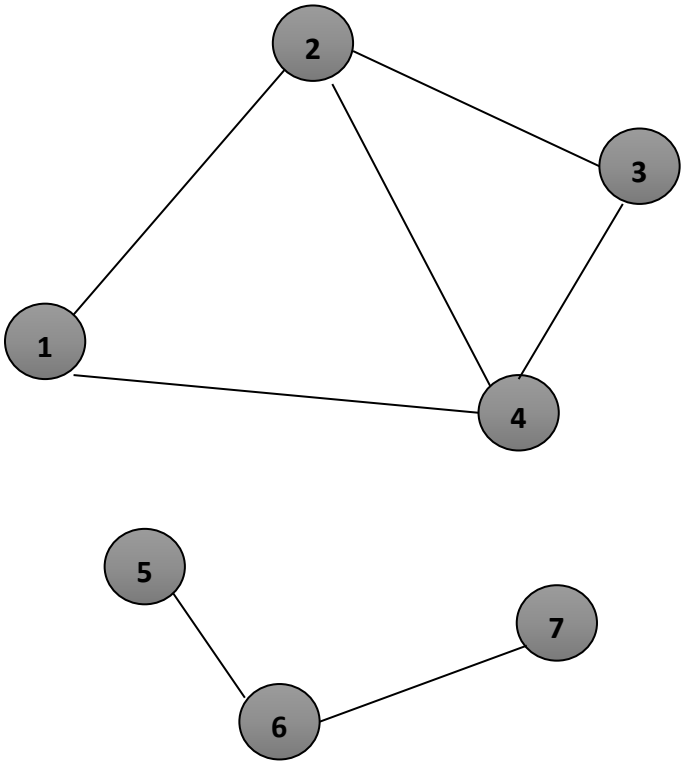


5. Să se verifice dacă un graf dat prin matricea de adiacenţă este **graf conex** si să se afiseze un mesaj corespunzător.

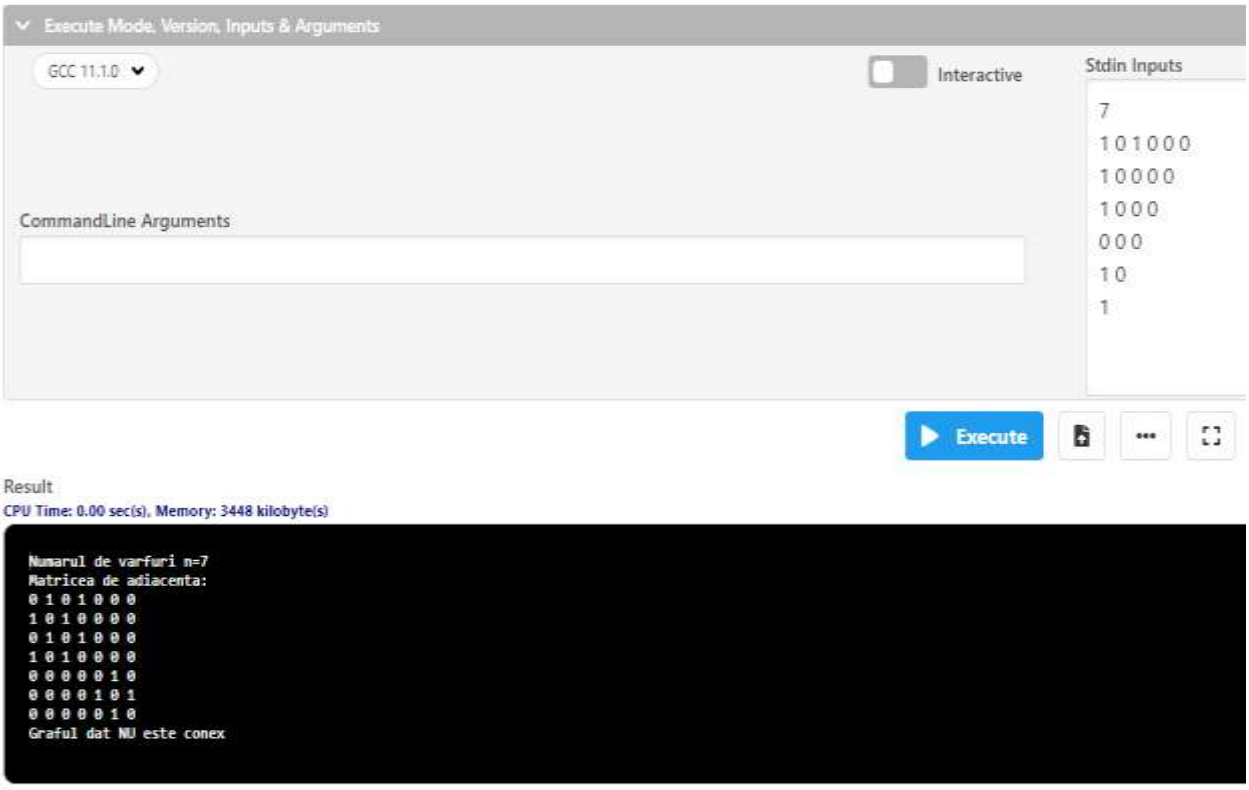
```
#include <iostream>
using namespace std;

int a[10][10], n, viz[10];
void citire()
{
int i,j;
for(i=1;i<n;i++)
        for(j=i+1;j<=n;j++)
        {
                cin>>a[i][j];
                a[j][i]=a[i][j];
        }
}
void parcurg(int x)
{       int i;
```

```
viz[x]=1;
for(i=1;i<=n;i++)
        if(a[x][i]&&viz[i]==0) parcurg(i);
}
int conex()
{
int i;
parcurg(1);
for(i=1; i<=n; i++)
        if(viz[i]==0)   return 0;
return 1;
}
int main()
{
cout<<"Numarul de varfuri n=";
cin>>n; cout<<n;
citire();
cout<<"\nMatricea de adiacenta:\n";
for(int i=1;i<=n;i++){
   for(int j=1;j<=n;j++)    cout<<a[i][j]<<" ";
   cout<<'\n';
}
if(conex()==1)                  cout<<"Graful dat este conex";
else            cout<<"Graful dat NU este conex";
   return 0;
}
```

Să considerăm următorul graf neorientat:



Executia programului folosind compilatorul online https://www.jdoodle.com/online-compiler-c++/:



Probleme propuse spre rezolvare

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

2. Fie graful neorientat G=(X,U), unde numărul de noduri este 7 si U={(1,2), (2,4), (2,7), (3,6), (3,7), (4,6), (4,7). (6,7)}. Sirul de noduri 7, 3, 6, 7, 4, 2, 7 este:
a) lant neelementar
b) lant compus
c) ciclu
d) ciclu elementar

3. Fie graful neorientat G=(X,U), unde numărul de noduri este 7 si U={(1,2), (2,4), (2,7), (3,6), (3,7), (4,6), (4,7). (6,7)}. Câte noduri cu grad impar sunt în graful G:
a) 3
b) 6
c) 5
d) 4

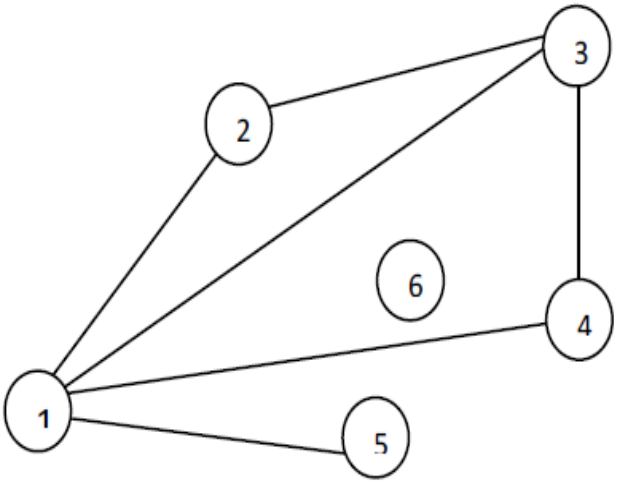
4. Scrieti răspunsul pentru fiecare dintre cerintele următoare.
a) Ce reprezintă un subgraf?
b) Desenati un graf bipartit care să aibă gradele nodurilor: 1, 1, 2, 2, 2, 3, 3
c) Scrieti o functie prin care să se verifice dacă un graf contine noduri izolate.
d) Care este numărul total de grafuri neorientate cu n noduri?

5. Fie graful neorientat G=(X,U), unde numărul de noduri este 8 si U={(2,5), (3,5), (3,6), (5,8), (6,7), (7,8)}. Care sunt nodurile care nu apartin nici unui ciclu?
a) 2, 3, 5
b) 1, 2, 4
c) 3, 7, 8
d) 1, 4, 7

6. Care dintre secventele următoare reprezintă sirul gradelor nodurilor unui graf complet?
a) 4 4 4 4
b) 2 2 2 2
c) 5 5 5 5 5 5
d) 3 3 3 3 3

Laborator 5

7. Scrieti răspunsul pentru fiecare dintre cerintele următoare.
a) Ce reprezintă un graf partial?
b) Pentru graful G=(X, U) din figură scrieti: matricea de adiacentă si nodul de grad maxim
c) Scrieti functia de parcurgere în lătime a unui graf.
d) Care este numărul de muchii al unui graf complet?
e) Scrieti o functie care afisează toate nodurile de grad maxim dintr-un graf neorientat.



8. Determinati vecinii unui varf al unui graf orientat.

9. Determinati gradele exterioare si interioare ale varfurilor unui graf, gradul exterior minim, gradul exterior maxim, gradul interior minim si gradul interior maxim.

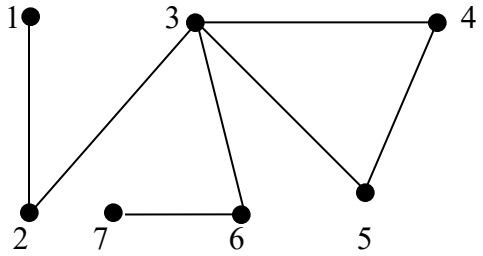
10. Verificati daca un graf este simetric/antisimetric.

11. Se citeste matricea de adiacenta a unui graf orientat. Sa se afiseze toate nodurile pentru care $d^+(x)=d^-(x)$ (gradul exterior este egal cu gradul interior). Pentru un nod x citit de la tastatura sa se afiseze toate nodurile adiacente cu acestea.

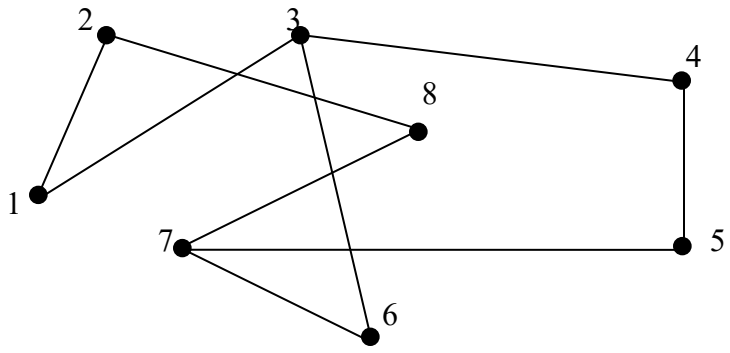
12. Sa se verifice daca o secventa de noduri data reprezinta un drum elementar sau ne-elementar intr-un graf orientat. Numarul de noduri, matricea de adiacenta, si secventa de noduri se citesc de la tastatura.

13. Descrieti in pseudocod sau in C++ algoritmii de cautare in adancime si in latime in grafuri. Parcurgeti in adancime, respectiv in latime urmatoarele grafuri:

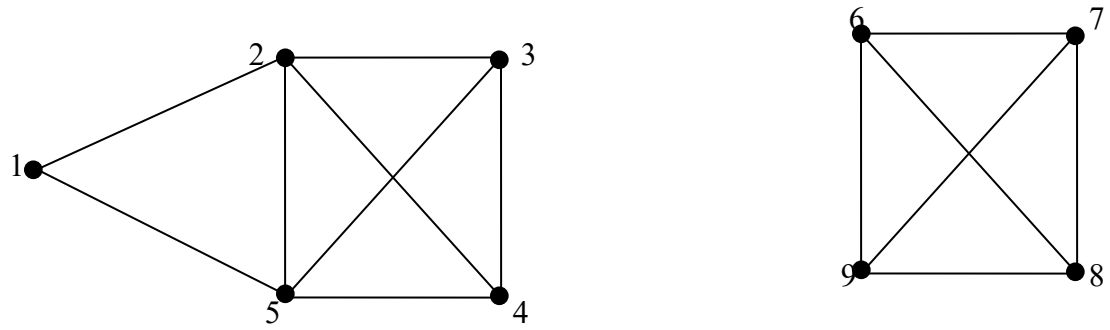
a) Graf neorientat



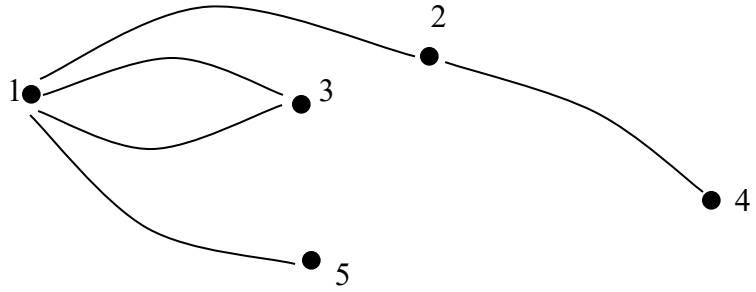
b) Graf neorientat



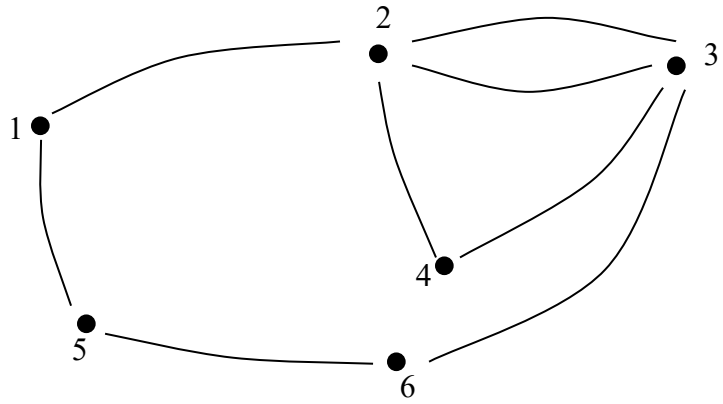
c) Graf neconex



d) Graf orientat



e) Graf orientat



f) Graf neorientat conex