```cpp
#include <fstream>
using namespace std; // #412
// Se dă lista muchiilor unui graf neorientat. Să se afiseze matricea de adiacentă a grafului.

ifstream cin("adiacenta.in");
ofstream cout("adiacenta.out");

int a[101][101];
int main()
{
    int n,m,x,y;
    cin>>n>>m; // numar noduri si numar muchii
    for(int i=1;i<=m;i++)
    {
        cin>>x>>y; // extremitatile unei muchii
        a[x][y]=a[y][x]=1;
    }
    for(int i=1;i<=n;i++) // afisare matricea de adiacenta
    {
      for(int j=1;j<=n;j++) cout<<a[i][j]<<' ';
       cout<<endl;
     }
    return 0;
}
```

C++

Fig.1

```cpp
#include <fstream>
using namespace std; // 413
ifstream cin("adiacenta1.in");
ofstream cout("adiacenta1.out");
int a[101][101];
//Se dă lista muchiilor unui graf neorientat. Să se afișeze matricea de adiacență a grafului.
int main()
{
    int n=0,m,x,y;
    // nu se cunosc numarul de noduri si muchii
    while(cin>>x>>y) // citim cate o muchie in mod repetat
    {
        a[x][y]=a[y][x]=1; // plasare 1 in matricea de adiacenta
        if(x>n) n=x;
        if(y>n) n=y; // determinam valoarea etichetei maxima.
    }
// eticheta maxima este si egala cu numarul de noduri din graf
    for(int i=1;i<=n;i++)
    {
      for(int j=1;j<=n;j++) cout<<a[i][j]<<' ';
       cout<<endl;
    }
    return 0;
}
```

C++

Fig.2

```cpp
#include <fstream>
using namespace std;// 414
ifstream cin("listavecini.in");
ofstream cout("listavecini.out");
int a[101][101];
// Se dă lista muchiilor unui graf neorientat. Să se afișeze, pentru fiecare vârf al grafului, lista vecinilor săi.
int main()
{
    int n=0,m,x,y;
    cin>>n;
    while(cin>>x>>y)
        a[x][y]=a[y][x]=1;

    for(int i=1; i<=n; i++) //linia nodului i
        for(int j=1; j<=n; j++) // numaram cate noduri j sunt vecine cu nodul i
            a[i][0]=a[i][0]+a[i][j]; // plasare in pozitia a[i][0]

    for(int i=1; i<=n; i++)
    {
        cout<<a[i][0]<<' '; // cati vecini are nodul i
        for(int j=1; j<=n; j++)
            if(a[i][j]==1)  cout<<j<<' '; // lista vecinilor j ai lui i
        cout<<endl;

    }
    return 0;
}
```

C++

Fig.3

```cpp
#include <fstream>
using namespace std; // 416
ifstream cin("grade.in");
ofstream cout("grade.out");
// Se dă lista muchiilor unui graf neorientat. Să se afișeze gradul fiecărui vârf.
int a[101][101];
int main()
{
    int n=0,m,x,y;
    cin>>n;
    while(cin>>x>>y)
        a[x][y]=a[y][x]=1;

    for(int i=1; i<=n; i++) //linia nodului i
        for(int j=1; j<=n; j++) // a[i][j] este 1 daca j este vecin al lui i
            a[i][0]=a[i][0]+a[i][j];// numaram vecinii lui i in a[i][0]

    for(int i=1; i<=n; i++)
          cout<<a[i][0]<<' ';

    return 0;
}
```

C++

Fig.4

```C++
#include <iostream>
using namespace std;// 2707
int a[102][102], n;
// Dându-se o matrice de numere întregi cu n linii și n coloane, să se verifice dacă este sau nu matrice de adiacență
// asociată unui graf neorientat.
int main()
{
    int ok=1, i, j;
    cin>>n;
    for(i=1; i<=n; i++)
        for(j=1; j<=n; j++)
        {
            cin >> a[i][j];
            if(i==j && a[i][j]!=0)  // pe DP trebuie sa avem valoarea 0
                ok=0;
        }
    for(i=1; i<n; i++)
        for(j=i+1; j<=n; j++)
        {
            if(a[i][j]<0 || a[i][j]>1) // In matrice putem avea doar valorile 0 si 1
                ok=0;
            if(a[i][j]!=a[j][i])
                ok = 0;
        }
    cout<<ok;
    return 0;
}
```

Fig.5

```C++
#include <fstream>
using namespace std;// 4068
ifstream cin ("gradek.in");
ofstream cout ("gradek.out");
int n, a[101][101], k, p;
// Se dă lista muchiilor unui graf neorientat și un nod k. Să se determine nodurile din graf care au gradul egal
// cu gradul nodului k.
int main()
{
    int x, y, i;
    cin>>n>>k;
    while(cin>>x>>y) // pentru fiecare muchie
    {
        if(!a[x][y] && !a[y][x]) // daca nu a fost citita inca
        {
            a[x][y]=a[y][x]=1;// plasare 1
            a[x][0]++;// gradul lui x
            a[y][0]++; // gradul lui y
        }
    }
    for(i=1; i<=n; ++i) // pentru fiecare nod i
            if(a[i][0]==a[k][0]) // verificam daca are gradul egal cu al nodului k
                p++;
    cout<<p<<endl; // cite noduri au gradul egal cu al nodului k
    for(i=1; i<=n; ++i)
            if(a[i][0]==a[k][0])
                cout<<i<<" "; // afisare etichetele nodurilor respective
    return 0;
}
```

Fig.6

```cpp
#include <iostream>
#include <fstream>
using namespace std; // 430

ifstream fin("izolate.in");
ofstream fout("izolate.out");

int n , a[105][105], ok[105];
// Se dă lista muchiilor unui graf neorientat. Să se afișeze vârfurile izolate ale grafului.
int main()
{
    int i , j;
    fin >> n;
    while(fin >> i >> j)
    {
        a[i][j] = a[j][i] = 1;
    }

    for(int i = 1 ;  i <= n ; i++)
    {
        ok[i] = 0;  // numaram cite noduri j sunt vecine cu nodul i
        for(int j = 1; j <= n ; ++j)
            if(a[i][j] == 1)
                ok[i] ++; // gradul nodului i
    }

    int nr_v = 0;
    for(int i = 1 ;  i <= n ; i++)
        if(ok[i] == 0) // i este nod izolat
                nr_v++;

    fout << nr_v << " "; // numar de noduri izolate
    for(int i = 1 ;  i <= n ; i++)
        if(ok[i] == 0)
            fout << i << " "; // etichetele nodurilor izolate

    return 0;
}
```

Fig.7

```cpp
#include <fstream>
using namespace std; // 4060
ifstream fin("gradk.in");
ofstream fout("gradk.out");
int a[101][101], n, k, p;
// Se dă un graf neorientat cu n vârfuri și un număr natural k. Să se afișeze vârfurile din graf care au gradul
// egal cu k.
int main()
{
    int x, y, i;
    fin>>n>>k;
    while(fin>>x>>y)
    {
        if(!a[x][y] && !a[y][x])
        {
            a[x][y]=a[y][x]=1;
            a[x][0]++;
            a[y][0]++; // crestem gradul extremitatilor x si y
        }
    }
    for(i=1; i<=n; i++)
            if(a[i][0]==k) // nodul i are gradul k
                p++;
    if(!p)
    {
        fout<<"NU EXISTA";
        return 0;
    }
    fout<<p<<' ';
    for(i=1; i<=n; i++)
            if(a[i][0]==k)
                fout<<i<<" ";
    return 0;
}
```

Fig.8

```cpp
#include <iostream>
#include <fstream>
using namespace std;// 417
ifstream fin("gradmax.in");
ofstream fout("gradmax.out");
int n , a[105][105], g[105];
// Se dă lista muchiilor unui graf neorientat. Să se afișeze vârfurile de grad maxim.
int main()
{
    int i , j;
    fin >> n;
    while(fin >> i >> j)
    {
        a[i][j] = a[j][i] = 1;
    }
    for(int i = 1 ;  i <= n ; i++){
        g[i] = 0;
        for(int j = 1; j <= n ; ++j)
            if(a[i][j] == 1)
                g[i] ++;  // gradul nodului i
    }
    for(int i = 1 ;  i <= n ; i++)
        cout << g[i] << " ";
    int gmax = 0 , nrv = 1;
    for(int i = 1 ;  i <= n ; i++)
        if(g[i] > gmax) // daca nodul i are gradul mai mare decit cel anterior
            gmax = g[i], nrv = 1; // actualizam gmax si nrv
        else
            if(g[i] == gmax)
                nrv ++; // numaram cate noduri au grad maxim

    fout << nrv << " ";
    for(int i = 1 ;  i <= n ; i++)
        if(g[i] == gmax)
            fout << i << " ";

    return 0;
}
```

Fig. 9

```cpp
//#include <fstream>
#include <iostream>
using namespace std; // 4070
//Se dă lista muchiilor unui graf neorientat cu n noduri, etichetate de la 1 la n, m muchii și un număr k.
// Din acest graf se elimină toate nodurile etichetate cu multipli ai lui k. Să se determine câte muchii
// va avea subgraful obtinut.

int main ()
{
  int n,m,w[105][105]={0},s=0,x,y,k;
  cin>>n>>m>>k;
  for(int i=1;i<=m;i++)
  {
    cin>>x>>y;
    w[x][y]=1;
    w[y][x]=1;
  }
  for(int i=1;i<=n;i++)
    for(int j=1;j<=n;j++)
       if(i%k==0)
           w[i][j]=w[j][i]=0;
  for(int i=1;i<=n;i++)
    for(int j=1;j<=n;j++)
        s=s+w[i][j];
  cout<<s/2;
}
```

C++

Fig. 10

```cpp
#include <fstream>
using namespace std;// 431
ifstream f("graf_complet.in");
ofstream g("graf_complet.out");
//  Se dau mai multe grafuri neorientate, prin matricea de adiacență. Să se verifice despre fiecare graf dacă
//  este complet.
int G, n;
int A[51][51];
bool ok;
int main()
{
    f>> G;
    for( ; G; --G )
    {
        f>> n; // numar noduri pentru graful curent
        ok = 1;
        for( int i = 1; i <= n; ++i )
            for( int j = 1; j <= n; ++j )
                f>> A[i][j]; // matricea de adiacenta a grafului curent
        for( int i = 1; i <= n && ok; ++i )
            for( int j = i + 1; j <= n && ok; ++j )
                if( !A[i][j] || !A[j][i] )// verificam daca exista valoarea 0 deasupra DP
                    ok = 0;
        g << (ok ? "DA\n" : "NU\n");
    }
    return 0;
}
```

C++

Fig. 11