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

//Verificarea daca un graf dat este sau nu un arbore

ifstream f("date.in");

int k,n,m,varf,nrc;
int viz[100],cc[100],tata[100],s[100],urm[100];
int a[100][100];

void viziteaza(int x)
{
    cc[x]=nrc;
}

void creareMatrice (int n, int m, int a[20][100])
{
    int x,y;
    for(int i=1; i<=m; i++)
    {
        f>>x>>y;
        a[x][y]++;
        if(x!=y)a[y][x]++;
    }
}

void df_recursiv (int x)
{
    viziteaza(x);
    viz[x]=1;
    for(int y=1; y<=n; y++)
    {
        if(a[x][y]>=1 && viz[y]==0)
        {
            tata[y]=x;
            df_recursiv(y);
        }
    }
}

void comp_conexe_neorientat ()
{
    int i;
    nrc=0;
    for(i=1; i<=n; i++)
        cc[i]=0;
    for(i=1; i<=n; i++)
        if(cc[i]==0)
        {
            nrc++;
```

```
            df_recursiv(i);
        }
}

void Arbore()
{
    if(nrc == 1 && m == n-1)
        cout<<"Graful este arbore!";
    else
        cout <<"Graful nu este arbore";
}

int main()
{
    f>>n;//nr de noduri
    f>>m;//nr de muchii

    creareMatrice(n,m,a);

    comp_conexe_neorientat();

    Arbore();

    return 0;

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

//Verificarea daca un graf orientat dat este sau nu quasi-tare conex.

ifstream f("date.in");

int n,m;
int d[100][100];
int a[100][100];
void creareMatrice(int n, int m, int a[100][100])
{
    int x,y;
    for(int i = 1; i <= m; i++)
    {
        f>>x>>y;
        a[x][y]++;
        //pt neorientat
    }
}

void afisareMatrice(int n, int a[100][100])
{
    for(int i=1; i <= n; i++)
    {
        for(int j=1; j<=n; j++)
            cout<<a[i][j]<<' ';
```

```
        cout<<'\n';
    }
}

void ROY_WARSHALL()
{
    int i,j,k;

    for(i = 1 ; i<=n; i++)
        for(j = 1 ; j<=n; j++)
            d[i][j]=a[i][j];
    for(k = 1 ; k<=n; k++)
        for(i = 1 ; i<=n; i++)
            for(j = 1 ; j<=n; j++)
                d[i][j] = d[i][j] | d[i][k] & d[k][j];

}

int quasiTareConex()
{
    int i,j;
    bool ok;
    for(i=1;i<=n;i++)
    {
        ok= true;
        for(j=1;j<=n;j++)
            if(i!=j && !d[i][j])
        {
            ok=false;
            break;
        }
        if(ok) return 1;
    }
    return 0;
}

int main()
{
    f>>n>>m;
    creareMatrice(n,m,a);
    afisareMatrice(n,a);
    cout<<'\n';
    ROY_WARSHALL();
    afisareMatrice(n,d);

    bool q = quasiTareConex();
    if(q)
        cout<<"Graful este quasi-tare conex";
    else
        cout<<"Graful nu este quasi-tare conex";
    return 0;
}
#include <iostream>
#include <fstream>
```

```cpp
using namespace std;

//Verificarea daca un graf orientat dat este sau nu o arborescenta

ifstream f("date.in");

int n,m;
int d[100][100];
int a[100][100];
void creareMatrice(int n, int m, int a[100][100])
{
    int x,y;
    for(int i = 1; i <= m; i++)
    {
        f>>x>>y;
        a[x][y]++;
        //pt neorientat
    }
}

void afisareMatrice(int n, int a[100][100])
{
    for(int i=1; i <= n; i++)
    {
        for(int j=1; j<=n; j++)
            cout<<a[i][j]<<' ';
        cout<<'\n';
    }
}

void ROY_WARSHALL()
{
    int i,j,k;

    for(i = 1 ; i<=n; i++)
        for(j = 1 ; j<=n; j++)
            d[i][j]=a[i][j];
    for(k = 1 ; k<=n; k++)
        for(i = 1 ; i<=n; i++)
            for(j = 1 ; j<=n; j++)
                d[i][j] = d[i][j] | d[i][k] & d[k][j];

}

int quasiTareConex()
{
    int i,j;
    bool ok;
    for(i=1;i<=n;i++)
    {
        ok= true;
        for(j=1;j<=n;j++)
            if(i!=j && !d[i][j])
        {
```

```
            ok=false;
            break;
        }
        if(ok) return 1;
    }
    return 0;
}

int main()
{
    f>>n>>m;
    creareMatrice(n,m,a);

    ROY_WARSHALL();

    bool q = quasiTareConex();
    if(q && m==n-1)
        cout<<"Graful este o arborescenrta";
    else
        cout<<"Graful nu este o arborescenrta";
    return 0;
}
```