#include<bits/stdc++.h>

using namespace std;

class Graph{

int no\_of\_nodes=0;

int no\_of\_edges=0;

bool visited\_nodes[50];

int adj\_matrix[50][50];

public:

Graph(int v){

no\_of\_nodes=v;

for(int i=0;i<no\_of\_nodes;i++)

{

// cout<<"for i = "<<i<<endl;

visited\_nodes[i]=false;

for(int j=0;j<no\_of\_nodes;j++){

adj\_matrix[i][j]=0;

}

}

}

void insert\_edges(){

int x,y;

cout<<"Enter number of edges = ";

cin>>no\_of\_edges;

for(int i=0;i<no\_of\_edges;i++){

cout<<"Enter edge in form of (x,y): ";

cin>>x>>y;

adj\_matrix[x][y]=adj\_matrix[y][x]=1;

}

}

void BFS(int starting\_node)

{

queue<int> q;

visited\_nodes[starting\_node]=true;

q.push(starting\_node);

BFSRecursive(q);

}

void BFSRecursive(queue<int> &q)

{

if(q.empty())

{

return ;

}

int front\_node=q.front();

q.pop();

cout<<front\_node<<" ";

// Add Adj node of fornt node in queue

for(int i=0;i<no\_of\_nodes;i++)

{

if(adj\_matrix[front\_node][i]==1 && !visited\_nodes[i])

{

visited\_nodes[i]=true;

q.push(i);

}

}

BFSRecursive(q);

}

void DFSRecursive(Graph &gr,int v,vector<bool> &visited\_nodes){

visited\_nodes[v]=true;

cout<<v<<" ";

for(int i=0;i<no\_of\_nodes;i++){

if(adj\_matrix[v][i]==1 && !visited\_nodes[i]){

DFSRecursive(gr,i,visited\_nodes);

}

}

}

};

int main()

{

int ver,s;

cout<<"Enter number of nodes = ";

cin>>ver;

Graph g(ver);

vector<bool> visited\_nodes(ver,false);

g.insert\_edges();

cout<<"Enter starting node : ";

cin>>s;

cout<<"BFS:";

g.BFS(s);

cout<<endl;

cout<<"DFS:";

g.DFSRecursive(g,s,visited\_nodes);

}