#include <iostream>

#include <vector>

#include <list>

#include <queue>

#include <stack>

#include <algorithm>

using namespace std;

class Graph {

private:

int numVertices;

vector<list<int>> adjList;

public:

Graph(int vertices) : numVertices(vertices) {

// Initialize adjacency list

adjList.resize(numVertices);

}

// Add an edge between vertices u and v

void addEdge(int u, int v) {

adjList[u].push\_back(v);

adjList[v].push\_back(u); // For undirected graph

}

// Perform BFS traversal starting from vertex startVertex

void bfsTraversal(int startVertex) {

vector<bool> visited(numVertices, false);

queue<int> q;

visited[startVertex] = true;

q.push(startVertex);

cout << "BFS Traversal starting from vertex " << startVertex << ": ";

while (!q.empty()) {

int currentVertex = q.front();

q.pop();

cout << currentVertex << " ";

for (int adjacentVertex : adjList[currentVertex]) {

if (!visited[adjacentVertex]) {

visited[adjacentVertex] = true;

q.push(adjacentVertex);

}

}

}

cout << endl;

}

// Perform DFS traversal starting from vertex startVertex

void dfsTraversal(int startVertex) {

vector<bool> visited(numVertices, false);

stack<int> s;

visited[startVertex] = true;

s.push(startVertex);

cout << "DFS Traversal starting from vertex " << startVertex << ": ";

while (!s.empty()) {

int currentVertex = s.top();

s.pop();

cout << currentVertex << " ";

// Visit all adjacent vertices

for (int adjacentVertex : adjList[currentVertex]) {

if (!visited[adjacentVertex]) {

visited[adjacentVertex] = true;

s.push(adjacentVertex);

}

}

}

cout << endl;

}

};

int main() {

int numVertices, numEdges;

cout << "Enter the number of vertices in the graph: ";

cin >> numVertices;

// Create a graph with user-specified number of vertices

Graph graph(numVertices);

cout << "Enter the number of edges in the graph: ";

cin >> numEdges;

cout << "Enter the edges (vertex pairs separated by space):" << endl;

for (int i = 0; i < numEdges; ++i) {

int u, v;

cin >> u >> v;

graph.addEdge(u, v);

}

int startVertex;

cout << "Enter the starting vertex for traversal: ";

cin >> startVertex;

// Perform BFS traversal

graph.bfsTraversal(startVertex);

// Perform DFS traversal

graph.dfsTraversal(startVertex);

return 0;

}