#include<iostream>

#include<stack>

#include<queue>

using namespace std;

class node {

public:

int data;

node\* next;

};

class Graph {

public:

stack<node\*> st;

queue<node\*> qu;

node\* head[20];

int no, visited[20];

void AdjacencyList();

void insertEdge(int, int);

void DFS();

void BFS();

};

void Graph::AdjacencyList() {

int i;

node\* f;

for (i = 0; i < no; i++) {

cout << endl;

f = head[i];

while (f) {

cout << "\t" << f->data;

f = f->next;

}

}

cout << endl;

}

void Graph::insertEdge(int v1, int v2) {

node\* n = new node;

n->data = v2;

n->next = NULL;

node\* f = head[v1];

if (f == NULL) {

head[v1] = n;

} else {

while (f->next)

f = f->next;

f->next = n;

}

// For undirected graph, insert edge in reverse direction

node\* m = new node;

m->data = v1;

m->next = NULL;

f = head[v2];

if (f == NULL) {

head[v2] = m;

} else {

while (f->next)

f = f->next;

f->next = m;

}

}

void Graph::DFS() {

node\* v, \* f;

while (!st.empty()) {

f = st.top();

st.pop();

if (visited[f->data] == 0) {

visited[f->data] = 1;

cout << " " << f->data;

v = head[f->data];

while (v) {

if (visited[v->data] == 0)

st.push(v);

v = v->next;

}

}

}

}

void Graph::BFS() {

node\* v, \* f;

while (!qu.empty()) {

f = qu.front();

qu.pop();

if (visited[f->data] == 0) {

visited[f->data] = 1;

cout << " " << f->data;

v = head[f->data];

while (v) {

if (visited[v->data] == 0)

qu.push(v);

v = v->next;

}

}

}

}

int main() {

Graph g;

int v, v1, v2;

char ch;

cout << "\nEnter the number of vertices in the graph: ";

cin >> g.no;

for (int i = 0; i < g.no; i++) {

g.head[i] = NULL;

g.visited[i] = 0;

}

cout << "\nNow you have to enter the edges (vertices labeled from 0 to " << g.no - 1 << "):\n";

do {

cout << "\nEnter edge (v1, v2): ";

cin >> v1 >> v2;

g.insertEdge(v1, v2);

cout << "\nDo you want to add more edges (y/n)? ";

cin >> ch;

} while (ch == 'y');

cout << "\nThe adjacency list of the graph is:\n";

g.AdjacencyList();

// DFS

cout << "\nEnter the starting vertex for DFS: ";

cin >> v;

g.st.push(g.head[v]);

cout << "\nThe DFS traversal of the graph is:\n";

g.DFS();

cout << endl;

for (int i = 0; i < g.no; i++)

g.visited[i] = 0;

// BFS

cout << "\nEnter the starting vertex for BFS: ";

cin >> v;

g.qu.push(g.head[v]);

cout << "\nThe BFS traversal of the graph is:\n";

g.BFS();

cout << endl;

return 0;

}