#include <iostream>

#include <cstring>

#include <cstdlib>

using namespace std;

class Vertex {

public:

char name[15];

int visit;

Vertex\* next;

};

Vertex\* lmark[20];

Vertex\* newVertex(char nm[15]) {

Vertex\* p = new Vertex();

if (p == NULL) {

cout << "Memory allocation failed!" << endl;

exit(1);

}

strcpy(p->name, nm);

p->visit = 0;

p->next = NULL;

return p;

}

int search(char x[15], int n) {

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

if (strcmp(lmark[i]->name, x) == 0)

return i;

}

return -1;

}

void DFS(char s[15], int n) {

Vertex\* temp;

char stack[10][15], x[15];

int top = -1, i;

cout << endl << "DFS of the given graph is" << endl;

top++;

strcpy(stack[top], s);

while (top != -1) {

strcpy(x, stack[top]);

top--;

i = search(x, n);

if (lmark[i]->visit == 0) {

cout << lmark[i]->name << endl;

lmark[i]->visit = 1;

}

else

continue;

temp = lmark[i]->next;

while (temp != NULL) {

i = search(temp->name, n);

if (lmark[i]->visit == 0) {

top++;

strcpy(stack[top], temp->name);

}

temp = temp->next;

}

}

}

void BFS(char s[15], int n) {

Vertex\* temp;

char Queue[10][15], x[15];

int front = -1, rear = -1, i;

cout << endl << "BFS of the given graph is" << endl;

if (rear == -1)

front++;

rear++;

strcpy(Queue[rear], s);

while (rear != -1) {

strcpy(x, Queue[front]);

front++;

if (front > rear)

front = rear = -1;

i = search(x, n);

if (lmark[i]->visit == 0) {

cout << lmark[i]->name << endl;

lmark[i]->visit = 1;

}

else

continue;

temp = lmark[i]->next;

while (temp != NULL) {

i = search(temp->name, n);

if (lmark[i]->visit == 0) {

if (rear == 0)

front = 0;

rear++;

strcpy(Queue[rear], temp->name);

}

temp = temp->next;

}

}

}

int main() {

int n, e, i, i1, i2, choice;

char v1[15], v2[15], s[15], nm[15];

Vertex\* p, \* q, \* temp;

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

cin >> n;

for (i = 0; i < n; i++) {

cout << "Enter the name of vertex " << i + 1 << ": ";

cin >> nm;

p = newVertex(nm);

lmark[i] = p;

}

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

cin >> e;

for (i = 0; i < e; i++) {

cout << "Enter edge " << i + 1 << ": ";

cin >> v1 >> v2;

i1 = search(v1, n);

i2 = search(v2, n);

if (i1 == -1 || i2 == -1) {

cout << "Invalid edge. Please enter valid vertices." << endl;

i--;

continue;

}

p = newVertex(v2);

q = newVertex(v1);

temp = lmark[i1];

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = p;

temp = lmark[i2];

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = q;

}

cout << endl << "Adjacency list is" << endl;

for (i = 0; i < n; i++) {

temp = lmark[i];

while (temp != NULL) {

cout << temp->name << "->";

temp = temp->next;

}

cout << endl;

}

cout << endl << "Enter the starting node: ";

cin >> s;

cout << endl;

while (true) {

cout << "\nMenu:\n 1. DFS\n 2. BFS\n 3. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

for (i = 0; i < n; i++) {

lmark[i]->visit = 0;

}

DFS(s, n);

break;

case 2:

for (i = 0; i < n; i++) {

lmark[i]->visit = 0;

}

BFS(s, n);

break;

case 3:

exit(0);

}

}

return 0;

}