/\* Anushka Surendra Unhale

SE B 65

\*/

//bfs dfs

#include<iostream>

#include <stack>

#include <queue>

using namespace std;

class graph

{

stack<int> s;

queue<int> Q;

public:

int n;

string vertices[10];

int weights[10][10];

int visited[10];

graph()

{

for(int i=0;i<10;i++)

{

for(int j=0;j<10;j++)

{

weights[i][j]=0;

}

}

}

void readGraph()

{

char ans;

cout<<"Enter no of locations: ";

cin>>n;

cout<<"Enter the locations:\n";

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

{

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

cin>>vertices[i];

}

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

{

for(int j=0;j<n;j++)

{

if(i!=j)

{

cout<<"Is there a direct path from "<<vertices[i]<<" to "<<vertices[j]<<"? (y/n): ";

cin>>ans;

if(ans=='y' || ans=='Y')

{

weights[i][j]=1;

/\*

cout<<"Enter the distance of the path(in metres): ";

cin>>weights[i][j];\*/

}

}

}

}

}

void makeVisitedZero()

{

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

{

visited[i]=0;

}

}

void dfs\_r(int i)

{

visited[i]=1;

cout<<"->"<<vertices[i];

for(int j=0;j<n;j++)

{

if(weights[i][j]!=0)

{

if(visited[j]==0)

dfs\_r(j);

}

}

}

void dfs\_nr(int i)

{

s.push(i);

while(!s.empty())

{

int v=s.top();

s.pop();

if(visited[v]==0)

{

cout<<"->"<<vertices[v];

visited[v]=1;

for(int j=0;j<n;j++)

{

if(weights[v][j]!=0)

s.push(j);

}

}

}

}

void bfs(int j)

{

visited[j]=1;

Q.push(j);

while(!Q.empty())

{

int i=Q.front();

cout<<"->"<<vertices[i];

Q.pop();

for(int j=0;j<n;j++)

{

if(weights[i][j]!=0)

{

if(visited[j]==0)

{

Q.push(j);

visited[j]=1;

}

}

}

}

}

void printVertices()

{

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

{

cout<<vertices[i]<<" : "<<i<<endl;

}

}

};

int main()

{

graph G;

G.readGraph();

int ch=1;

int v;

while(ch)

{

cout<<"\n~~~~~~~~~~~~~~~~MENU~~~~~~~~~~~~~\n";

cout<<"1.Recursive DFS Traversal\n 2.Non-recursive DFS Traversal\n 3.BFS Traversal 0.EXIT\n ENter your choice: ";

cin>>ch;

switch(ch)

{

case 0: cout<<"BYE!";

break;

case 1: G.makeVisitedZero();

cout<<"INDICES:\n";

G.printVertices();

cout<<"Enter the index of vertex from which traversal is to be started: ";

cin>>v;

G.dfs\_r(v);

break;

case 2: G.makeVisitedZero();

cout<<"INDICES:\n";

G.printVertices();

cout<<"Enter the index of vertex from which traversal is to be started: ";

cin>>v;

G.dfs\_nr(v);

break;

case 3: G.makeVisitedZero();

cout<<"INDICES:\n";

G.printVertices();

cout<<"Enter the index of vertex from which traversal is to be started: ";

cin>>v;

G.bfs(v);

break;

default:cout<<"\n\nENTER A VALID CHOICE!\n\n";

}

}

}