#include <iostream>

#include <string.h>

#define MAX 10

using namespace std;

class djs

{

public:

int i, j, k, a, b, u, v, n, ne;

int min, mincost, cost[MAX][MAX], process[MAX];

void shortestpath(int , int);

void displayGraph();

void create();

djs()

{

for(int i=1; i<=MAX; i++)

for(int j=1; j<=MAX; j++)

cost[i][j]=999;

}//constructor

};

void djs::create()

{

int ch, wt;

cout<<"No. of Vertices in a graph :: ";

cin>>n;

for(int i=1; i<n; i++) //Vertices/Nodes

{

for(int j=i+1; j<=n; j++) //Adjacency nodes

{

cout<<"Does edge is present between "<<i<<" and "<<j<<"?(1/0)- ";

cin>>ch;

if(ch==1)

{

cout<<"Enter the weight: ";

cin>>wt;

cost[i][j]=cost[j][i]=wt;

}

} //outer for - j

} //outer for - i

} //create

void djs::displayGraph()

{

int i,j;

cout<<"The entered Graph is: \n";

for(i=1; i<=n; i++)

{

for(j=1;j<=n;j++)

{

cout<< "\t" <<cost[i][j];

}

cout<< "\n";

}

} //display Graph

void djs::shortestpath(int s, int t)

{

int newdist, dist[MAX], processed[MAX], preced[MAX], curr,i,k,dc,smalldist;

int p,j=0; //for path

int path[MAX];

for(int i=1; i<=n; i++) //initialize

{

processed[i]=preced[i]=0;

dist[i]=999; //initialize distance to infinity

}

processed[s]=1; //start from source

dist[s]=0;

curr=s;

//WHILE LOOP OF SHORTEST PATH STARTS HERE

while(curr!=t) //till not at destination

{

smalldist=999;

dc=dist[curr]; //distance of current node-dc

for(i=1; i<=n; i++)//scan all vertices for finding shortest distance from curr

{

if(processed[i]==0)

{

newdist=dc+cost[curr][i];

if(newdist<dist[i])

{

dist[i]=newdist;

preced[i]=curr;

}

if(dist[i]<smalldist)

{

smalldist=dist[i];

k=i;

}

}//end of if not processed

}//end of for

curr=k;

processed[curr]=1;

}//end of while loop of shortest path Ends Here

//FRAME THE PATH OF SHORTEST DISTANCE FROM SOURCE TO DESTINATION

p=t;

path[j++]=p;

while(p!=s)

{

p=preced[p];

path[j++]=p;

}

cout<<"\n Shortest path: ";

for(i=j-1; i>=0; i--)

{

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

}

cout<<"\n"<<"Shortest distance: "<<dist[t]<<"\n";

} //shortest path

int main()

{

int s,t;

djs obj; //Constructor will be called

obj.create();

obj.displayGraph();

cout<<"\n Enter start and target vertices: ";

cin>>s;

cin>>t;

obj.shortestpath(s,t);

return 0;

}

1.The very first step is to mark all nodes as unvisited.

2.Mark the picked starting node with a current distance of 0 and the rest nodes with infinity.

3.Now, fix the starting node as the current node.

4.For the current node, analyse all of its unvisited neighbours and measure their distances by

adding the current distance of the current node to the weight of the edge that connects the

neighbour node and current node.

5.Compare the recently measured distance with the current distance assigned to the neighbouring

node and make it as the new current distance of the neighbouring node.

6.After that, consider all of the unvisited neighbours of the current node, mark the current node

as visited.

7.If the destination node has been marked visited then stop,the algorithm has ended.

8.Else, choose the unvisited node that is marked with the least distance, fix it as the new current

node, and repeat the process again from step 4.