#include <iostream>

using namespace std;

class MyGraph{

public:

int am[100][100]; // Adjacency Matrix

void addEdge(int n); // Function to Add Edge

void display(int n, int dist[]); // Function to Display shortest paths

void shortestPath(int n, int source); // Dijkstra's Shortest path algorithm;

};

//\*\*\*\*\*\*\*\*\*\* GRAPH INPUT FUNCTION \*\*\*\*\*\*\*\*\*\*

void MyGraph::addEdge(int n){

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

cout<<"Enter Vertex-1, Vertex-2 and weight of edge between Vertex-1,Vertex-2"<<endl;

int u,v,w;

cout<<"Vertex-1: ";

cin>>u;

cout<<"Vertex-2: ";

cin>>v;

cout<<"Weight: ";

cin>>w;

cout<<endl;

am[u][v] = w;

am[v][u] = w;

}

}

//\*\*\*\*\*\*\*\*\*\* DIJKSTRA'S SHORTEST PATH ALGORITHM \*\*\*\*\*\*\*\*\*\*

void MyGraph::shortestPath(int n, int source){

int dist[n]; // Array to store shortest distances

bool sptSet[n]; // Array to store whether a node is visited or not

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

dist[i] = 32767; // Initializing distance array to infinity

sptSet[i] = false; // Initializing visited node array to false

}

dist[source] = 0; // Distance from source is 0

int edge\_count;

cout<<"Enter the Number of edges: "<<endl;

cin>>edge\_count;

addEdge(edge\_count);

for(int count = 0; count<edge\_count-1; count++){

int u;

int mini = 32767;

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

if(sptSet[i] == false && dist[i] < mini){

mini = dist[i];

u = i;

}

}

sptSet[u] = true; // minimum distance node is marked visited

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

if(sptSet[v] == false && am[u][v] && dist[u] < mini && dist[v]>dist[u]+am[u][v]){

dist[v] = dist[u] + am[u][v]; //calculating shortest path

}

}

}

display(n, dist); // displaying shortest paths

}

//\*\*\*\*\*\*\*\*\*\* DISPLAY SHORTEST PATH \*\*\*\*\*\*\*\*\*\*

void MyGraph::display(int n, int dist[]){

cout<<"Source:"<<"\t\t"<<"Shortest Distance"<<endl;

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

cout<<i<<"\t\t\t"<<dist[i]<<endl;

}

cout<<endl;

}

//\*\*\*\*\*\*\*\*\*\* MAIN FUNCTION \*\*\*\*\*\*\*\*\*\*

int main() {

MyGraph g;

int num,src;

cout<<"Enter the number of nodes of the graph: "<<endl;

cin>>num;

cout<<"Enter the source Node: "<<endl;

cin>>src;

g.shortestPath(num, src);

return 0;

}