//program to find the shortest path from a source, in cpp

#include<iostream>

#include<string>

#define INF 999

#define NUM\_OF\_VERTICES 6

using namespace std;

int minDist(int dist[],bool visited[]) //this function returns the shortest unvisited node from the dist array

{

int min = INF;

int index = 0;

for(int i=0;i<NUM\_OF\_VERTICES;i++)

{

if(dist[i]<min&&visited[i]!=true)

{

min = dist[i];

index = i;

}

}

return index;

}

void shortestPath(int src,int cost[][NUM\_OF\_VERTICES],int n)

{

int dist[n];//distance array which holds all the shortest distance from the source

bool visited[n]; //array that holds all the nodes that are visited or not

int optnode; //u as the optimal node can be taken as u

int opt\_dist; //the optimal distance from the source

string path[n];

//initialize the dist and visited arrays

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

{

dist[i] = INF;

visited[i] = false;

path[i] = to\_string(src);

}

dist[src] = 0;//sets the distance from the source node to source node as 0

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

{

int optnode = minDist(dist,visited); //calls the minDist funct which returns the minimum in the dist array

visited[optnode] = true;

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

{

if(!visited[j]&&cost[optnode][j]&&dist[j]>dist[optnode]+cost[optnode][j])// determines whether the distance array will be updated based on the optnode

{

dist[j] = dist[optnode]+cost[optnode][j];

path[j] = path[optnode] +"->"+ to\_string(j);

}

}

}

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

{

if(i!=src&&dist[i]==INF)

{

path[i] = "Unreachable";

}

}

cout<<"The shortest dist for each node to the source is "<<endl;

cout<<"Source\t\t"<<"Destination\t\t"<<"Cost\t\t"<<"Path\t\t"<<endl;

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

{

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

}

}

int main()

{

int cost\_adj[NUM\_OF\_VERTICES][NUM\_OF\_VERTICES] = {{0,50,45,10,0,0}, //input matrix

{0,0,10,15,0,0},

{0,0,0,0,30,0},

{20,0,0,0,15,0},

{0,20,35,0,0,0},

{0,0,0,0,3,0}

};

shortestPath(0,cost\_adj,NUM\_OF\_VERTICES);

return 0;

}

OUTPUT:

