/*Graph: Shortest Path Algorithm: Represent a given graph using adjacency matrix /adjacency list and find the shortest path using Dijkstra's algorithm (single source all destination).*/

```
#include<iostream>
using namespace std;

class graph
{
int g[20][20];
int e,v;
public:

void accept()
{
  int src,dest,cost,i,j;
  cout<<"\n Enter no. of vertices -";
  cin>>v;
  cout<<"\n Enter no. of edges ";
  cin>>e;
  for(i=0;i<v;i++)
  {
     for(j=0;j<v;j++)
     {
       g[i][j]=0;
     }
  }

  for(i=0;i<e;i++)
  {
     cout<<"\n Enter source and destination -";
     cin>>src>>dest;
     cout<<"\n Enter the cost of edges - ";
     cin>>cost;
     g[src][dest]=cost;
     g[dest][src]=cost;
}
}

void display()
{
  int i,j;
  for(i=0;i<v;i++)
  {
    cout<<endl;
  for(j=0;j<v;j++)
  {
    cout<<g[i][j]<<"\t";
  }
  }
}
```

```
void djikstra(int start)
{
  int r[20][20],mindst,next,cnt,i,j,visited[20],distance[20],from[20];

  for(i=0;i<v;i++)          //intialization of r[][]
  {
     for(j=0;j<v;j++)
     {
       if(g[i][j]==0)
        r[i][j]=9999;
       else
        r[i][j]=g[i][j];
     }
  }

  for(i=0;i<v;i++)      //intialization of visited[],distance[],from[]
  {
     visited[i]=0;
     from[i]=start;
     distance[i]=r[start][i];
  }
     distance[start]=0;
     visited[start]=1;
     cnt=v;
  while(cnt>0)
  {
    mindst=9999;
      for(i=0;i<v;i++)
      {
          if((mindst > distance[i] )&& visited[i]==0)
           {
              mindst=distance[i];
              next=i;
           }
      }

    visited[next]=1;
    for(i=0;i<v;i++)
      {
          if(visited[i]==0 && distance[i]>(mindst+r[next][i]))
          {
            distance[i]=mindst+r[next][i];
            from[i]=next;
           }
      }
        cnt--;
}

for(i=0;i<v;i++)
 {
cout<<"\n Distance of "<<i<<" from "<<start<<" is "<<distance[i]<<endl<<" path "<<i;
```

```
 j=i;
do
{
j=from[j];
cout<<"<-"<<j;;
 }while(j!=start);
}
}

};

int main()
{
  int s;
  graph g;
  g.accept();
  g.display();
  cout<<endl<<"Enter the starting vertex -";
  cin>>s;
  g.djikstra(s);
  cout<<endl;
  return 0;
}
```

Enter no. of vertices -5

Enter no. of edges 4

Enter source and destination -1 2

Enter the cost of edges - 2

Enter source and destination -1 3

Enter the cost of edges - 5

Enter source and destination -2 4

Enter the cost of edges - 6

Enter source and destination -1 4

Enter the cost of edges - 8

```
0    0    0    0    0
0    0    2    5    8
0    2    0    0    6
0    5    0    0    0
0    8    6    0    0
```

Enter the starting vertex -1

Distance of 0 from 1 is 9999
path 0<-1
Distance of 1 from 1 is 0
path 1<-1
Distance of 2 from 1 is 2
path 2<-1
Distance of 3 from 1 is 5
path 3<-1
Distance of 4 from 1 is 8
path 4<-1

...Program finished with exit code 0
Press ENTER to exit console.