```
1: //                                        Assignment no-8
2: // Name- Prerana Rajesh Gajare      Class-SEIT      RollNo-SI41
3: /*PROBLEM STATEMENT:-
4:     Represent a graph of the city using an adjacency matrix /adjacency
   list. Nodes should represent the various
5:     landmarks and links should represent the distance between them. Find
   the shortest path using Dijkstra's
6:     algorithm from a single source to all destinations.
7: */
8: //Source Code:-
9: #include <iostream>
10: using namespace std;
11:
12: //Class graph
13: class graph
14: {
15:     //declaring variables
16:     int source,dest,weight,v,e;
17:     int mat[20][20];
18:
19:     public:
20:         //declaring meathod
21:         void getdata();
22:         void display_matrix();
23:         void display(int [],int [],int);
24:         int min_key(int [],bool []);
25:         void dijkstra();
26: };
27:
28: //To accept input using getdata function
29: void graph::getdata()
30: {
31:     //Accepting the number of vertices and edges in the graph
32:     cout<<"Enter the total no. of vertices : ";
33:     cin>>v;
34:     cout<<"Enter the total no. of edges : ";
35:     cin>>e;
36:     //Creating a matrix of vertices and initialise all element as 0
37:     for(int i=0;i<v;i++)
38:     {
39:         for(int j=0;j<v;j++)
40:         {
41:         mat[i][j]=0;
42:         }
43:     }
44:      //Accepting input
45:     for(int i=0;i<e;i++)
46:     {
47:         cout<<"\nEnter the source vertex :";
48:         cin>>source;
```

```
49:         cout<<"Enter the destination vertex :";
50:         cin>>dest;
51:         cout<<"Enter the weight of edge :";
52:         cin>>weight;
53:         mat[source][dest]=weight;
54:         mat[dest][source]=weight;
55:     }
56: }
57:
58: //To display the vetex,distance from vertex and path
59: void graph ::display(int dist[],int parent[],int src)
60: {
61:     int nodes[v]={0},n,e;
62:     cout<<"\nVertex\t\tDistance\tPath\n";
63:
64:     for(int i=0;i<v;i++)
65:     {
66:         cout<<i<<"\t\t"<<dist[i]<<"\t\t";
67:         e=i;
68:         n=0;
69:         while(e!=src)
70:         {
71:             nodes[n]=e;
72:             e=parent[e];
73:             n++;
74:         }
75:         nodes[n]=e;
76:         for(int j=n;j>0;j--)
77:         {
78:             cout<<nodes[j]<<"-";
79:         }
80:         cout<<nodes[0];
81:         cout<<"\n";
82:     }
83: }
84:
85: //to print the adjancy matrix
86: void graph ::display_matrix()
87: {
88:     cout<<"Adjancy matrix is :";
89:     cout<<"\n";
90:     for(int i=0;i<v;i++)
91:     {
92:
93:         for(int j=0;j<v;j++)
94:         {
95:             cout<<"\t"<<mat[i][j];
96:        }
97:        cout<<"\n";
98:     }
```

```
 99: }
100:
101: //To find the vertex with minimum key value, from the visit array of
     vertices not yet included in MST
102: int graph ::min_key(int a[],bool b[])
103: {
104:     int min_index=0;
105:     int max=999;
106:     for(int i=0;i<v;i++)
107:     {
108:         if(b[i]==false && a[i]<max)
109:         {
110:             max=a[i];
111:             min_index=i;
112:         }
113:     }
114:     return min_index;
115: }
116:
117: //To dijkstra algorithm
118: void graph ::dijkstra()
119: {
120:
121:     int parent[v],dist[v],s;//parent array to store parent of each node and
     distance array to store distance of node from parent node
122:     bool visit[v];  //to store the status of each node if it is visited or
     not
123:     for(int i=0;i<v;i++)
124:     {
125:         parent[i]=i;//initialise each node as vertex of itself
126:         dist[i]=999;//initialise  all nodes as infinity/999
127:         visit[i]=false;//and visit of all nodes as false
128:     }
129:     //Accepting source of MST
130:     cout<<"\nEnter the Start vertex :";
131:     cin>>s;
132:     source=s;
133:     dist[s]=0;
134:     for(int i=0;i<v;i++)
135:     {
136:         s=min_key(dist,visit);//find minimum distance from neighbouring
     nodes of source
137:         visit[s]=true;//and mark its index as visited(true) in visited array
138:         for(int j=0;j<v;j++)
139:         {
140:             if(mat[s][j]!=0 && visit[j]==false && dist[s]+mat[s][j]<dist[j])
141:             {
142:                 dist[j]=dist[s]+mat[s][j];
143:                 parent[j]=s;
144:             }
```

```
145:         }
146:     }
147:     display(dist,parent,source);//Calling display function
148: }
149:
150: int main()
151: {
152:     graph g;//object of class graph
153:     g.getdata();//Calling getdata function
154:     g.display_matrix();//Calling display_matrix function
155:     g.dijkstra();//Calling diskstar function
156:     return 0;
157: }
158:
```

C:\Users\Dell\OneDrive\Documents\PRERANA.CPP\P.CPP\Dijkstrar.exe

```
Enter the total no. of vertices : 5
Enter the total no. of edges : 7

Enter the source vertex :0
Enter the destination vertex :1
Enter the weight of edge :2

Enter the source vertex :0
Enter the destination vertex :2
Enter the weight of edge :9

Enter the source vertex :1
Enter the destination vertex :2
Enter the weight of edge :7

Enter the source vertex :2
Enter the destination vertex :4
Enter the weight of edge :5

Enter the source vertex :1
Enter the destination vertex :3
Enter the weight of edge :10

Enter the source vertex :3
Enter the destination vertex :2
Enter the weight of edge :6

Enter the source vertex :3
Enter the destination vertex :4
```

C:\Users\Dell\OneDrive\Documents\PRERANA.CPP\P.CPP\Dijkstrar.exe

```
Enter the destination vertex :2
Enter the weight of edge :6

Enter the source vertex :3
Enter the destination vertex :4
Enter the weight of edge :4
Adjancy matrix is :
        0       2       9       0       0
        2       0       7       10      0
        9       7       0       6       5
        0       10      6       0       4
        0       0       5       4       0

Enter the Start vertex :0

Vertex          Distance        Path
0               0               0
1               2               0-1
2               9               0-2
3               12              0-1-3
4               14              0-2-4

--------------------------------
Process exited after 54.97 seconds with return value 0
Press any key to continue . . .
```