```
1: //                                          Assignment no-
   7
2: // Name- Prerana Rajesh Gajare      Class-SEIT
   RollNo-SI41
3: /*PROBLEM STATEMENT:-
4:         Represent a graph of your college campus using
   adjacency list /adjacency matrix. Nodes should
   represent the
5:          various departments/institutes and links
   should represent the distance between them. Find
   minimum spanning tree
6:         a) Using Prim's algorithm.
7: */
8: //Source Code:-
9: #include <iostream>
10: using namespace std;
11:
12: //Class graph
13: class graph
14: {
15:     int mat[20][20];//initialise matrix[20][20]
16:     public:
17:         //declaring variables and meathod
18:         int source,dest,weight,v,e;
19:         int getdata();
20:         void add_edge(int ,int);
21:         void prim();
22:         int min_key(int [],bool []);
23:         void prim_display(int []);
24:         void display_matrix();
25:
26: };
27:
28: //To accept input using getdata function
29: int graph :: getdata()
30: {
31:     //Accepting the number of vertices and edges in
   the graph
```

```
32:     int i,j;
33:     cout<<"Enter total no of vertices :";
34:     cin>>v;
35:     cout<<"Enter total no of edges :";
36:     cin>>e;
37:     //Creating a matrix of vertices and initialise all
    element as 0
38:     for( i=0;i<v;i++)
39:     {
40:         for( j=0;j<v;j++)
41:         {
42:             mat[i][j]=0;
43:         }
44:     }
45:
46:      //Accepting input

47:     for( i=0;i<e;i++)
48:     {
49:         cout<<"Enter source vertex:";
50:         cin>>source;
51:         cout<<"Enter destination vertex";
52:         cin>>dest;
53:         cout<<"Enter the weight of tree edge:";
54:         cin>>weight;
55:         mat[source][dest]=weight;
56:         mat[dest][source]=weight;
57:
58:     }
59:
60: }
61:
62: //To print the adjancy matrix
63: void graph:: display_matrix()
64: {
65:     cout<<"Adjancy matrix is :";
66:     cout<<"\n";
```

```
67:     for(int i=0;i<v;i++)
68:     {
69:         for(int j=0;j<v;j++)
70:         {
71:             cout<<"\t"<<mat[i][j];
72:        }
73:        cout<<"\n";
74:     }
75: }
76:
77: //To find the vertex with minimum key value, from the
    visit array of vertices not yet included in MST
78: int graph :: min_key(int key [],bool visit[])
79: {
80:     int p;
81:     int max=999;
82:     int min_index=0;
83:     for(p=0;p<v;p++)
84:     {
85:         if(visit[p]==false && key[p] < max)
86:         {
87:             max=key[p];
88:             min_index=p;
89:         }
90:     }
91:     return min_index;
92: }
93:
94: //To display the prims minimum spanninfg tree
95: void graph :: prim_display(int parent[])
96: {
97:     cout<<"Minimum spaning tree:";
98:     cout<<"\nEdge\t\tWeight\n";
99:     for(int i=1;i<v;i++)
100:     {
101:
102:             cout<<parent[i]<<" ->
    "<<i<<"\t\t"<<mat[i][parent[i]]<<"\n";
```

```
103:
104:
105:     }
106: }
107:
108: void graph :: prim()
109: {
110:     int parent[v];//parent array to store parent of
     each node
111:     int key[v];//to store distance of node from parent
     node
112:     bool visit[v];//to store the status of each node
     if it is visited or not
113:     for(int i=0;i<v;i++)
114:     {
115:         key[i]=999;//initialise  all nodes as
     infinity/999
116:         visit[i]=false;//and visit of all nodes as
     false
117:
118:     }
119:     cout<<endl;
120:     //Accepting source of MST
121:     cout<<"\nSource vertex of MST :";
122:     cin>>source;
123:     key[source]=0;
124:     for(int a=0;a<v-1;a++)
125:     {
126:         int u= min_key(key,visit);//find minimum
     distance from neighbouring nodes of source
127:         visit[u]=true;//and mark its index as visited
     in visited array
128:         for(int j=0;j<v;j++)
129:         {
130:             if((mat[u][j]!=0) && visit[j] == false &&
     mat[u][j]<key[j])
131:             {
```

```
132:                 parent[j]=u;
133:                 key[j]=mat[u][j];
134:             }
135:         }
136:     }
137:     prim_display(parent);//calling display function
138: }
139:
140:
141: int main()
142: {
143:     graph g;//object of class graph
144:     g.getdata();//Calling getdata function
145:     g.display_matrix();//Calling display matrix
     function
146:
147:     g.prim();//Calling prim function
148: }
```

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

```
Enter total no of vertices :5
Enter total no of edges :7
Enter source vertex:0
Enter destination vertex1
Enter the weight of tree edge:2
Enter source vertex:0
Enter destination vertex2
Enter the weight of tree edge:9
Enter source vertex:1
Enter destination vertex2
Enter the weight of tree edge:7
Enter source vertex:2
Enter destination vertex4
Enter the weight of tree edge:5
Enter source vertex:1
Enter destination vertex3
Enter the weight of tree edge:10
Enter source vertex:3
Enter destination vertex2
Enter the weight of tree edge:6
Enter source vertex:3
Enter destination vertex4
Enter the weight of tree 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


Source vertex of MST :0
Minimum spaning tree:
Edge            Weight
0 -> 1          2
1 -> 2          7
4 -> 3          4
2 -> 4          5

--------------------------------
```