```
1: //                                          Assignment no-
   7(1)
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 Kruskal's algorithm.
7: */
8: //Source Code:-
9: #include <iostream>
10: using namespace std;
11:
12: //Class graph
13: class graph
14: {
15:     public:
16:         //declaring variables and meathod
17:         int source,dest,weight;
18:         int getdata();
19:         void display(graph*,int);
20:         void sorting(graph *[],int );
21:         int check(int,int *);
22:         graph *kruskal( graph*,int,int);
23: };
24:
25: //To check if each node has parent of itself or not.
26: int graph :: check(int v,int *parent)
27: {
28:     if(parent[v]==v)
29:     {
30:         return v;
31:     }
```

```
32:     return check(parent[v],parent);
33: }
34:
35:
36: graph* graph :: kruskal(graph *edge,int n,int e)
37: {
38:     //First sort the edges in the ascending order in
    terms of weight array
39:     for(int i=0;i<n-1;i++)
40:     {
41:         for(int j=0;j<n-i-1;j++)
42:         {
43:             if(edge[j].weight>edge[j+1].weight){
44:                 graph temp=edge[j];
45:                 edge[j]=edge[j+1];
46:                 edge[j+1]=temp;
47:             }
48:         }
49:     }
50:     //Memory allocation to store edge
51:     graph* MST=new graph[n-1];
52:     int parent[n];
53:     for(int i=0;i<n;i++)
54:     {
55:         parent[i]=i; //Parent array to store parent of
    each node
56:     }
57:     int count=0;
58:     int i=0;
59:     while(count!=n-1)
60:     {
61:         graph curredge=edge[i];
62:         int srcpar= check(curredge.source,parent);
63:         int destpar=check(curredge.dest,parent);
64:         if(srcpar!=destpar)
65:         {
66:             MST[count]=curredge;
```

```
            count++;
            parent[srcpar]=destpar;
        }
        i++;
    }
    cout<<"minimum spanning tree:"<<endl;
    cout<<"SOURCE"<<" "<<"DESTINATION"<<"
"<<"WEIGHT"<<endl;
    int cost=0;
    for(int i=0;i<n-1;i++)
    {
        cout<<MST[i].source<<" "<<MST[i].dest<<"
"<<MST[i].weight<<endl;
        cost=cost+MST[i].weight;
    }
    cout<<"MINIMUM COST: "<<cost;


}

int main()
{
    int s,w,d,v,e;
    int mat[20][20];//initialise matrix[20][20]
    //Accepting the number of vertices and edges in
the graph
    cout<<"Enter number of vertices:";
    cin>>v;
    cout<<"Enter no of edges :";
    cin>>e;
    for(int i=0;i<v;i++)
    {
        for(int j=0;j<v;j++)
        {
            mat[i][j]=0;
        }
    }
```

```
101:     //Allocating memory to store edge e in *edge in
     form of array
102:     graph *edge=new graph[e];
103:         for(int i=0;i<e;i++)
104:     {
105:         //Accepting input
106:         cout<<"Enter source vertex:";
107:         cin>>s;
108:         cout<<"Enter destination vertex";
109:         cin>>d;
110:         cout<<"Enter the weight of edge:";
111:         cin>>w;
112:         mat[s][d]=w;
113:         mat[d][s]=w;
114:         //Storing the value of source,destination and
     weight in edge array at i th index
115:         edge[i].source=s;
116:         edge[i].dest=d;
117:         edge[i].weight=w;
118:     }
119:     //To print the adjancy matrix
120:     cout<<"Adjancy matrix is :";
121:     cout<<"\n";
122:     for(int i=0;i<v;i++)
123:     {
124:         for(int j=0;j<v;j++)
125:         {
126:             cout<<"\t"<<mat[i][j];
127:        }
128:        cout<<"\n";
129:     }
130:
131:     graph g;//Creating object of class graph
132:     g.kruskal(edge,v,e);//Calling Kruskal function
133: }
```

C:\Users\Dell\OneDrive\Documents\PRERANA.CPP\P.CPP\kruskal0.exe

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

C:\Users\Dell\OneDrive\Documents\PRERANA.CPP\P.CPP\kruskal0.exe

```
Enter the weight of edge:5
Enter source vertex:1
Enter destination vertex3
Enter the weight of edge:10
Enter source vertex:3
Enter destination vertex2
Enter the weight of edge:6
Enter source vertex:3
Enter destination vertex4
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
minimum spanning tree:
SOURCE DESTINATION WEIGHT
0 1 2
2 4 5
1 2 7
1 3 10
MINIMUM COST: 24
--------------------------------
Process exited after 38.54 seconds with return value 0
Press any key to continue . . .
```