Name   : Resham Landge
Roll No : 2339

******************************Assignment No : 7**********************************

Title : Represent any real world graph using adjacency list /adjacency matrix find minimum spanning tree using Kruskal’s algorithm.

****

```
#include<iostream>
#define MAX 30                                          //define the size of MAX is 30

using namespace std;

struct edge                                             //to declare the stucture
{
        int u,v,w;
};

class edgelist
{
        public:
                edge data[MAX];;
                int n;

                friend class graph;
                edgelist()                              //to initialize the n is NULL
                {
                        n=0;
                }

                void sort();
                void print();
};

void edgelist::sort()                                   //this function sort the edges by weight
{
        int i,k;
        edge temp;

        for(i=1; i<n; i++)
                for(k=0; k<n-i; k++)
                        if(data[k].w > data[k+1].w)             //sorting logic
```

```
            {
                temp=data[k];
                data[k]=data[k+1];
                data[k+1]=temp;
            }
}

void edgelist::print()                  // this function display the MST of tree
{
    int i,cost=0;

    for(i=0; i<n; i++)
    {
        cout<<"\n\tEdge of "<<data[i].u<<" to "<<data[i].v<<" of Weight is : "<<data[i].w;
        cost=cost+data[i].w;
    }
    cout<<"\n\n\tCost of spapning tree = "<<cost<<"\n\n";
}

class graph                             //to declare the graph class
{
    public:
        int G[MAX][MAX],n;
        graph()
        {
            n=0;
        }

        void create();
        void kruskal(edgelist &span);
};

void graph::create()                    //create a graph
{
    int i,k;

    cout<<"\n\tEnter No. of vertices : ";
    cin>>n;

    cout<<"\n\tEnter the adjacency matrix : \n";
    for(i=0; i<n; i++)
        for(k=0; k<n; k++)
            cin>>G[i][k];
```

```
}

int find(int belong[],int vertexno);                //component no.of vertex
void combine(int belong[],int c1,int c2,int n);     //combining two components

int main()
{
    edgelist span;                                  //list of edges in the spanning tree
    graph g;

    g.create();
    g.kruskal(span);
    span.print();
}

void graph::kruskal(edgelist &span)
{
    int belong[MAX],i,k,no1,no2;
    edgelist list;                                  //all edges are stored in list

    for(i=1; i<n; i++)
        for(k=0; k<n; k++)
        {
            if(G[i][k] !=0)
            {
                list.data[list.n].u=i;
                list.data[list.n].v=k;
                list.data[list.n].w=G[i][k];
                list.n++;
            }
        }

    list.sort();
    for(i=0; i<n; i++)                      //initialize belong
        belong[i]=i;

    for(i=0; i<list.n; i++)                 //add edges of the graph to the sapnning tree
    {
        no1=find(belong,list.data[i].u);
        no2=find(belong,list.data[i].v);

        if(no1!=no2)
        {
            span.data[span.n]=list.data[i];
```

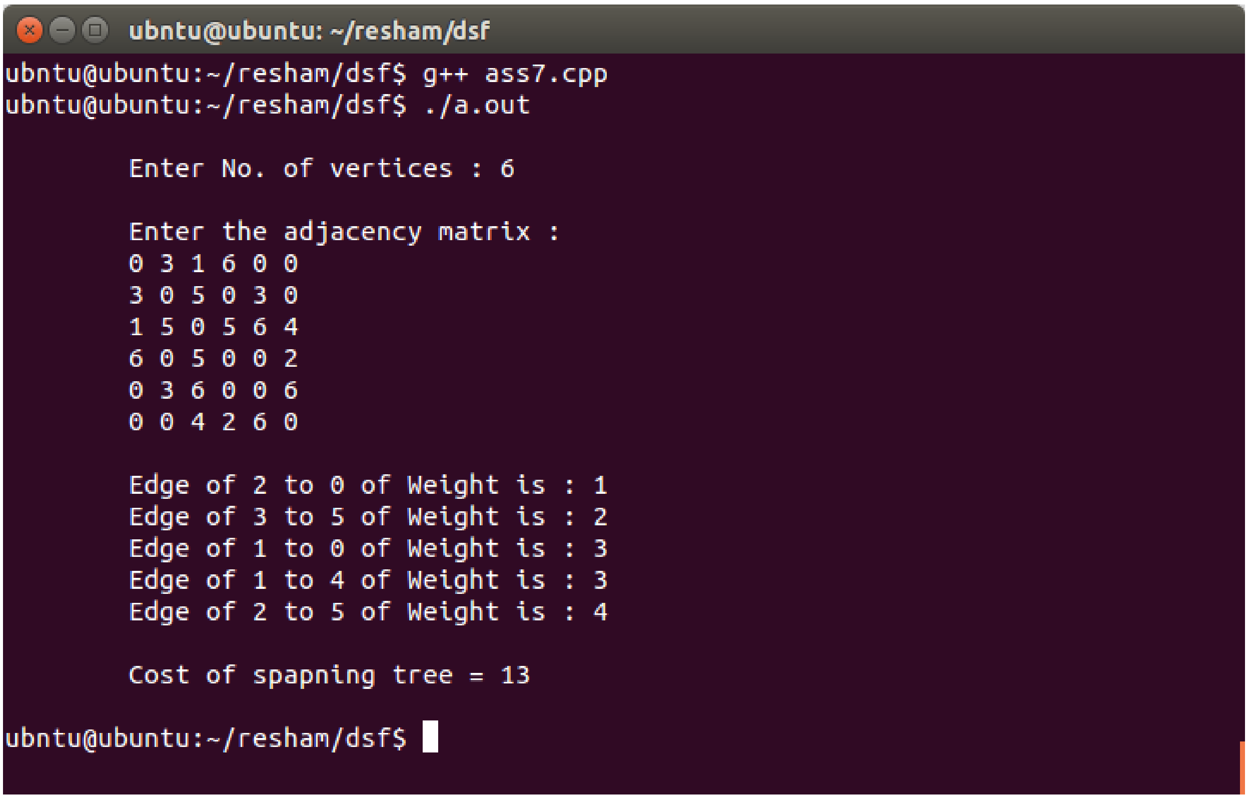
```
                    span.n=span.n+1;
                    combine(belong,no1,no2,n);
                }
        }
}

int find(int belong[],int vertexno)                    //return component number of a vertex
{
        return(belong[vertexno]);
}

void combine(int belong[],int c1,int c2,int n)        //merge tow component c1 and c2into
                                                       single comonent c1
{
        int i;

        for(i=0; i<n; i++)
                if(belong[i]==c2)
                        belong[i]=c1;         //component are represented in the array belong[]
}
```

Output :



```
ubntu@ubuntu:~/resham/dsf$ g++ ass7.cpp
ubntu@ubuntu:~/resham/dsf$ ./a.out

        Enter No. of vertices : 6

        Enter the adjacency matrix :
        0 3 1 6 0 0
        3 0 5 0 3 0
        1 5 0 5 6 4
        6 0 5 0 0 2
        0 3 6 0 0 6
        0 0 4 2 6 0

        Edge of 2 to 0 of Weight is : 1
        Edge of 3 to 5 of Weight is : 2
        Edge of 1 to 0 of Weight is : 3
        Edge of 1 to 4 of Weight is : 3
        Edge of 2 to 5 of Weight is : 4

        Cost of spapning tree = 13

ubntu@ubuntu:~/resham/dsf$ 
```