Name: Pratik Rameshwar Shinde

Roll No.: 2274

Class: SE-IT

Div: B-3

Subject: DSA LAB

Practical 7: Graph- Minimum Spanning Tree

Aim: Represent a graph of your college campus using adjacency list /adjacency matrix. Nodes should represent the various departments/institutes and links should represent the distance between them. Find minimum spanning treeb) Using Kruskal’s algorithm.

*****************************PROGRAM******************************

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


class MST

{

        int a[20][20],n,k; struct
        gr
        {
                int v1; int
        v2; int wt; }g[20];


        public:
                void accept(); void
                extract_edges(); void
                kruskal();
};


void MST::accept()

{

        int i,j;
         cout<<"\nEnter the no. of vertices: ";
cin>>n;


        cout<<"Enter adjacency matrix:\n";
```

```
    for(i=1;i<=n;i++)
             for(j=1;j<=n;j++)
                  cin>>a[i][j];
}

void MST::extract_edges()
{
      int i,j;

      for(i=1,k=0;i<=n;i++) for(j=i+1;j<=n;j++)
            if(a[i][j]!=0 )
                  {
                        g[k].v1=i; g[k].v2=j;
                        g[k++].wt=a[i][j]; }

      cout<<"\tSource\tSink\tWeight\n";

            for(i=0;i<k;i++) cout<<"\t"<<g[i].v1<<"\t"<<g[i].v2<<"\t"<<g[i].wt<<"\n";
}

void MST::kruskal()
{ gr temp,tree[20];
      int i,j,father[20]={0},sum=0,n1,n2,r1,r2;

      for(i=0;i<k;i++)
      {
    for(j=0;j<k-1;j++)
            {
                  if(g[j].wt>g[j+1].wt)
           {
                  temp=g[j+1];
                  g[j+1]=g[j]; g[j]=temp;
         }
            }
      }
```

```
        cout<<"\tSource\tSink\tWeight\n"; for(i=0;i<k;i++)
        cout<<"\t"<<g[i].v1<<"\t"<<g[i].v2<<"\t"<<g[i].wt<<"\n";

        for(i=0,j=0;i<k && j<n-1;i++)
        { n1=g[i].v1; n2=g[i].v2;

                        while(n1>0)
                        { r1=n1; n1=father[n1];
                        }

                        while(n2>0)
                        { r2=n2; n2=father[n2];
                        }

                        if(r1!=r2)
                        {
                                tree[j].v1=g[i].v1; tree[j].v2=g[i].v2;
                                tree[j++].wt=g[i].wt; sum+=g[i].wt;
                                father[r2]=r1;

                        }
        } cout<<"\nEdges in MST:\n\tSource\tSink\tWeight\n";

        for(i=0;i<j;i++) cout<<"\t"<<tree[i].v1<<"\t"<<tree[i].v2<<"\t"<<tree[i].wt<<"\n";

        cout<<"Total cost of MST: "<<sum<<"\n";
}

int main()
{ MST m;
        int ch;

        m.accept();
        m.extract_edges();
        m.kruskal();

        return 0;
```

```
}
*****************************OUTPUT*
*****************************


[admin@fedora ~]$ g++ hfbdsa7b.cpp
[admin@fedora ~]$ ./a.out

Enter the no. of vertices: 5   Enter
adjacency matrix:
0  1 2 2 0
1  0 0 0 2
2  0 0 1 0
2 0 1 0 1
    0        2 0 1 0
        Source          Sink     Weight
    1        2    1
        1         3         2
        1         4         2
        2         5         2
        3         4         1
        4         5         1
        Source          Sink     Weight
        1         2         1
        3         4         1
        4         5         1
        1         3         2
        1         4         2
        2         5         2

Edges in MST:
        Source          Sink     Weight
        1         2         1
        3         4         1
        4         5         1
        1         3         2 Total
cost of MST: 5
[admin@fedora ~]$
```