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 treea) Using Prim'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
                prims();
};

void MST::accept()
{
        int i,j;
        cout<<"\nEnter the no. of vertices: ";      cin>>n;

        cout<<"Enter adjacency matrix:\n";
    for(i=0;i<n;i++)
```

```
                for(j=0;j<n;j++)
                        cin>>a[i][j];
}

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

        for(i=0,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<<"Edges in the graph are:\n\tSource\tSink\tWeight\n";

                for(i=0;i<k;i++)
                cout<<"\t"<<char(g[i].v1+65)<<"\t"<<char(g[i].v2+65)<<"\t"<<g[i].wt<<"\
n"; }

void MST::prims()
{ int i,j,min_edge,visited[20]={0},sum=0,min,flag;

        visited[0]=1;
        for(i=0;i<k;i++)
        {
                min=1000; flag=0;
                for(j=0;j<k;j++)
                { if((visited[g[j].v1]==0&&visited[g[j].v2]==1)||
(visited[g[j].v1]==1&&visited[g[j].v2]==0)) if(g[j].wt<min)
                                { min=g[j].wt;
                                        min_edge=j;
                                        flag=1;
                                }
                }
                if(flag)
                { cout<<"\nEdge included
"<<char(g[min_edge].v1+65)<<"--"<<char(g[min_edge].v2+65)<<" with weight
```

```
"<<g[min_edge].wt;\ visited[g[min_edge].v1]=1;
                    visited[g[min_edge].v2]=1;
                    sum+=g[min_edge].wt;
                }
        }
        cout<<"\nTotal cost of MST: "<<sum<<"\n";
}

int main()
{ MST m;

        m.accept();
        m.extract_edges();
        m.prims();

        return 0;
} ******************************OUTPUT*********************************
```

```
[admin@fedora ~]$ g++ hfbdsa7a.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
Edges in the graph are:
        Source          Sink    Weight
        A       B       1
        A       C       2
        A       D       2
        B       E       2
        C       D       1
        D       E       1
```

Edge included A--B with weight 1

Edge included A--C with weight 2 Edge included C--D with weight 1

Edge included D--E with weight 1

Total cost of MST: 5

[admin@fedora ~]$