Practical No.: -9 Write a program to find Minimum-Cost Spanning Trees (Prim's & Kruskal's Algorithm).

A) Prim's Algorithm: -

```
#include<iostream>
#include<conio.h>
using namespace std;
int Near[50],cost[6][6],t[6][6],k,e,L=2,n,mc,i,j;
class prim1
{
public:
void getdata();
void putdata();
void prim2(int cost[6][6],int n,int t[6][6],int mc);
};
void prim1::getdata()
{
cout<<"\n\n Enter Number of Vertices:-";
cin>>n;
cout<<"\n Enter Matrix"<<n<<"X"<<n<<":-\n";
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
        cin>>cost[i][j];
}
void prim1::putdata()
{
for(i=1;i<n;i++)
cout<<"\n Edge"<<i<<"==>"<<t[i][1]<<"<======>"<<t[i][2];
}
void prim1::prim2(int cost[6][6],int n,int t[6][6],int mc)
{
int s,f,p,m,p1;
k=1;
mc=cost[1][2];
t[1][1]=k;
t[1][2]=L;
for(int i=1;i<=n;i++)
{
if(cost[i][L]<cost[i][k])
```

```
        Near[i]=L;
else
        Near[i]=k;
}
Near[k]=0;
Near[L]=0;
for(i=2;i<n;i++)
{
p=i;
m=i;
p1=cost[m][Near[m]];
for(;p<=n;p++)
{
if(Near[p]!=0&&cost[p][Near[p]]<=p1)
{
p1=cost[p][Near[p]];
j=p;
if(p==i&&i==n-1)
        break;
}
}
t[i][1]=j;
t[i][2]=Near[j];
mc=mc+cost[j][Near[j]];
Near[j]=0;
m++;
for(int k=1;k<=n;k++)
        if(Near[k]!=0&&cost[k][Near[k]]>cost[k][j])
                Near[k]=j;
}
if(mc>=500)
cout<<"\n\n Spanning Tree";
else
{
cout<<"\n\n Spanning Tree";
cout<<"\n\n Mincost:- "<<mc<<endl;
}
}
void main()
{
```

```
prim1 p;
//clrscr();
p.getdata();
p.prim2(cost,n,t,mc);
p.putdata();
getch();
}
```

/*OUTPUT*/

```
Enter Number of Vertices:-6
Enter Matrix6X6:-
0 10 500 30 45 500
10 0 50 500 40 25
500 50 0 500 35 15
30 500 500 0 500 20
45 40 35 500 0 55
500 25 15 20 55 0
 Spanning Tree
 Mincost:- 105
 Edge1==>1<======>2
 Edge2==>6<======>2
 Edge3==>3<======>6
 Edge4==>4<======>6
 Edge5==>5<======>3
```

B. Kruskal's Algorithm: -

```
#include<iostream>
#include<conio.h>
using namespace std;
class krushal
{
int n,i,noe,cost[100][4],tree[10][10],sets[100][10],top[100];
public:
void getdata();
void algorithm();
int find(int);
void display();
}obj;
void krushal::getdata()
{
int w;
cout<<"Enter the no. of nodes in the undirected weighted graph:-";
cin>>n;
noe=0;
cout<<"Enter the weights for the following edges:-\n";
for(int i=1;i<=n;i++)
{
for(int j=i+1;j<=n;j++)
{
cout<<"<"<<i<<","<<j<<">::";
cin>>w;
if(w!=0)
{
noe++;
cost[noe][1]=i;
cost[noe][2]=j;
cost[noe][3]=w;
}
}
}
}
void krushal::algorithm()
{
int mincost=0;
```

```
for(int i=1;i<=n;i++)
{
sets[i][1]=i;
top[i]=1;
}
cout<<"The algorithm starts:-\n";
for(i=1;i<=noe;i++)
{
int p1=find(cost[i][1]);
int p2=find(cost[i][2]);
if(p1!=p2)
{
cout<<"The edge included in the tree is:-"<<"<"<<cost[i][1]<<",";
cout<<cost[i][2]<<">"<<endl;
tree[cost[i][1]][cost[i][2]]=cost[i][3];
tree[cost[i][2]][cost[i][1]]=cost[i][3];
mincost=mincost+cost[i][3];
for(int j=1;j<=top[p2];j++)
{
top[p1]++;
sets[p1][top[p1]]=sets[p2][j];
}
top[p2]=0;
}
}
cout<<"Mincost:-"<<mincost;
}
int krushal::find( int n)
{
for(int i=1;i<=noe;i++)
        for(int j=1;j<=top[i];j++)
                if(n==sets[i][j])
return i;
return -1;
}
void main()
{
//clrscr();
obj.getdata();
obj.algorithm();
```

```
getch();
}
```

/*OUTPUT*/

```
Enter the no. of nodes in the undirected weighted graph:-5
Enter the weights for the following edges:-
<1,2>::10
<1,3>::15
<1,4>::20
<1,5>::25
<2,3>::30
<2,4>::35
<2,5>::40
<3,4>::45
<3,5>::20
<4,5>::55
The algorithm starts:-
The edge included in the tree is:-<1,2>
The edge included in the tree is:-<1,3>
The edge included in the tree is:-<1,4>
The edge included in the tree is:-<1,5>
Mincost:-70
```