Program 1:

Design and implement C/C++ program to find Minimum Cost Spanning Tree of a given connected undirected graph using Kruskal's algorithm.

Algorithm:

Algorithm : Kruskal(G)
// Kruskal's algorithm for constructing a minimum spanning tree
//Input: A weighted connected graph G=(V,E)
//Output: E_T,the set of edges composing a minimum spanning tree of G
{
Sort E in non decreasing order of the edge weights $w(e_{i1})<=\ldots\ldots>= w(e_{i\,|E|})$
$E_T \leftarrow \varnothing$; ecounter $\leftarrow$ 0 //Initialize the set of tree edges and its size
k $\leftarrow$ 0 //initialize the number of processed edges
while ecounter <|V|-1 do
{
k $\leftarrow$ k+1
if $E_T U \{e_{ik}\}$ is acyclic
$E_T \leftarrow E_T U \{e_{ik}\}$; ecounter $\leftarrow$ ecounter+1
}
return E_T
}

Code:

```
#include<stdio.h>
int ne=1, min_cost=0;
void main()
{
int n,i,j,min,a,u,b,v,cost[20][20],parent[20];
printf("Enter the number of vertices: ");
scanf("%d", &n);
printf("\nEnter the cost matrix: \n");
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
scanf("%d", &cost[i][j]);
for(i=1;i<=n;i++)
parent[i]=0;
```

```
printf("\n The edges of spanning tree are\n");
while(ne<n)
{
min=999;
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
if(cost[i][j]<min)
{
min=cost[i][j];
a=u=i;
b=v=j;
}
}
}
while(parent[u])
u=parent[u];
while(parent[v])
v=parent[v];
if(u!=v)
{
printf("Edge %d\t(%d->%d)=%d\n",ne++,a,b,min);
min_cost=min_cost+min;
parent[v]=u;
}
cost[a][b]=cost[a][b]=999;
}
printf("\n Minimum cost=%d\n",min_cost);
}
```

Output:

```
Enter the number of vertices: 6

Enter the cost matrix:
23 34 56 78 34 12
11 33 78 899 89 34
222 44 66 87 98 444
11 33 44 76 54 22
14 56 78 89 90 54
12 45 67 89 65 46

 The edges of spanning tree are
Edge 1  (2->1)=11
Edge 2  (4->1)=11
Edge 3  (1->6)=12
Edge 4  (5->1)=14
Edge 5  (3->2)=44

 Minimum cost=92
```

Program 2:

Design and implement C/C++ Program to find Minimum Cost Spanning Tree of a given connected undirected graph using Prim's algorithm.

Algorithm:

Algorithm : Prim(G)
// Prim's algorithm for constructing a minimum spanning tree
//Input: A weighted connected graph G=(V,E)
//Output: E_T, the set of edges composing a minimum spanning tree of G
{
    $V_T \leftarrow \{v_0\}$ //the set of tree vertices can be initialized with any vertex
    $E_T \leftarrow \varnothing$

    for i ← 0 to |V| - 1 do
        find a minimum-weight edge e* = (v*, u*) among all the edges (v, u)
        such that v is in V_T and u is in $V-V_T$
        $V_T \leftarrow V_T \cup \{u^*\}$
        $E_T \leftarrow E_T \cup \{e^*\}$

    return E_T
}

Code:

```
#include<stdio.h>
int ne=1,min_cost=0;
void main()
{
int n,i,j,min,cost[20][20],a,u,b,v,source,visited[20];
printf("Enter the number of nodes: ");
scanf("%d",&n);
printf("Enter the cost matrix:\n");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
scanf("%d",&cost[i][j]);
```

```
}
}
for(i=1;i<=n;i++)
visited[i]=0;
printf("Enter the root node: ");
scanf("%d",&source);
visited[source]=1;
printf("\n Minimum cost spanning tree is\n");
while(ne<n)
{
min=999;
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
if(cost[i][j]<min)
if(visited[i]==0)
continue;
else
{
min=cost[i][j];
a=u=i;
b=v=j;
}
}
}
if(visited[u]==0||visited[v]==0)
{
printf("\nEdge %d\t(%d->%d)=%d\n",ne++,a,b,min);
min_cost=min_cost+min;
```

```
visited[b]=1;
}
cost[a][b]=cost[b][a]=999;
}
printf("\nMinimum cost=%d\n",min_cost);
}
```

Output:

```
Enter the number of nodes: 4
Enter the cost matrix:
23 567 1 4
34 3 67 999
2 4 65 34
34 67 98 12
Enter the root node: 1

 Minimum cost spanning tree is

Edge 1  (1->3)=1

Edge 2  (1->4)=4

Edge 3  (3->2)=4

Minimum cost=9
```