

1. Write a c program for Minimum Spanning Tree.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define MAX 100
4 typedef struct {
5     int u, v, w;
6 } Edge;
7 typedef struct {
8     Edge edges[MAX];
9     int n;
10 } Edgelist;
11 Edgelist elist;
12 int parent[MAX];
13 Edgelist spanlist;
14 int find(int v) {
15     if (parent[v] == v)
16         return v;
17     return find(parent[v]);
18 }
19 void union_set(int u, int v) {
20     parent[u] = v;
21 }
22 void kruskal(int n) {
23     int i, j, u, v;
24     for (i = 0; i < n; i++) {
25         parent[i] = i;
26     }
27     spanlist.n = 0;
28     for (i = 0; i < elist.n; i++) {
29         u = find(elist.edges[i].u);
30         v = find(elist.edges[i].v);
31         if (u != v) {
32             spanlist.edges[spanlist.n] = elist.edges[i];
33             spanlist.n = spanlist.n + 1;
34             union_set(u, v);
35         }
36     }
37 }
```

```
/tmp/S0cJEF40cc.o
Enter the number of vertices: 4
Enter the number of edges: 5
Enter edge 1 (u, v, weight): 0 1 10
Enter edge 2 (u, v, weight): 0 2 6
Enter edge 3 (u, v, weight): 0 3 5
Enter edge 4 (u, v, weight): 1 3 15
Enter edge 5 (u, v, weight): 2 3 4
Edges in the Minimum Spanning Tree:
2 - 3 : 4
0 - 3 : 5
0 - 1 : 10
Total cost of Minimum Spanning Tree: 19

=== Code Execution Successful ===
```

```
34         union_set(u, v);
35     }
36 }
37 }
38 void sort() {
39     int i, j;
40     Edge temp;
41     for (i = 1; i < elist.n; i++)
42         for (j = 0; j < elist.n - 1; j++)
43             if (elist.edges[j].w > elist.edges[j + 1].w) {
44                 temp = elist.edges[j];
45                 elist.edges[j] = elist.edges[j + 1];
46                 elist.edges[j + 1] = temp;
47             }
48 }
49 void print() {
50     int i, cost = 0;
51     printf("Edges in the Minimum Spanning Tree:\n");
52     for (i = 0; i < spanlist.n; i++) {
53         printf("%d - %d : %d\n", spanlist.edges[i].u, spanlist.edges[i].v,
54             spanlist.edges[i].w);
55         cost += spanlist.edges[i].w;
56     }
57     printf("Total cost of Minimum Spanning Tree: %d\n", cost);
58 }
59 int main() {
60     int n, e, i;
61     printf("Enter the number of vertices: ");
62     scanf("%d", &n);
63     printf("Enter the number of edges: ");
64     scanf("%d", &e);
65     elist.n = e;
66     for (i = 0; i < e; i++) {
67         printf("Enter edge %d (u, v, weight): ", i + 1);
68     }
```

```
/tmp/S0cJEF40cc.o
Enter the number of vertices: 4
Enter the number of edges: 5
Enter edge 1 (u, v, weight): 0 1 10
Enter edge 2 (u, v, weight): 0 2 6
Enter edge 3 (u, v, weight): 0 3 5
Enter edge 4 (u, v, weight): 1 3 15
Enter edge 5 (u, v, weight): 2 3 4
Edges in the Minimum Spanning Tree:
2 - 3 : 4
0 - 3 : 5
0 - 1 : 10
Total cost of Minimum Spanning Tree: 19

=== Code Execution Successful ===
```

```
60     printf("Enter the number of vertices: ");
61     scanf("%d", &n);
62     printf("Enter the number of edges: ");
63     scanf("%d", &e);
64     elist.n = e;
65     for (i = 0; i < e; i++) {
66         printf("Enter edge %d (u, v, weight): ", i + 1);
67         scanf("%d%d%d", &elist.edges[i].u, &elist.edges[i].v, &elist.edges[i].w);
68     }
69     sort();
70     kruskal(n);
71     print();
72     return 0;
73 }
```

2. Write a C program for Prims Algorithm.

```
1 #include <stdio.h>
2 #include <stdbool.h>
3 #include <limits.h>
4 #define MAX 100
5 int n;
6 int graph[MAX][MAX];
7 int minKey(int key[], bool mstSet[]) {
8     int min = INT_MAX, min_index;
9     for (int v = 0; v < n; v++)
10         if (mstSet[v] == false && key[v] < min)
11             min = key[v], min_index = v;
12     return min_index;
13 }
14 void printMST(int parent[]) {
15     int total_cost = 0;
16     printf("Edge Weight\n");
17     for (int i = 1; i < n; i++) {
18         printf("%d - %d %d\n", parent[i], i, graph[i][parent[i]]);
19         total_cost += graph[i][parent[i]];
20     }
21     printf("Total cost of Minimum Spanning Tree: %d\n", total_cost);
22 }
23 void primMST() {
24     int parent[MAX];
25     int key[MAX];
26     bool mstSet[MAX];
27
28     for (int i = 0; i < n; i++) {
29         key[i] = INT_MAX;
30         mstSet[i] = false;
31     }
32     key[0] = 0;
33     parent[0] = -1;
34     for (int count = 0; count < n - 1; count++) {
```

```
/tmp/9Dbmffffp2S.o
Enter the number of vertices: 5
Enter the adjacency matrix:
0 2 0 6 0
2 0 3 8 5
0 3 0 0 7
6 8 0 0 9
0 5 7 9 0
Edge Weight
0 - 1 2
1 - 2 3
0 - 3 6
1 - 4 5
Total cost of Minimum Spanning Tree: 16

=== Code Execution Successful ===
```

```
31     }
32     key[0] = 0;
33     parent[0] = -1;
34     for (int count = 0; count < n - 1; count++) {
35         int u = minKey(key, mstSet);
36         mstSet[u] = true;
37         for (int v = 0; v < n; v++)
38             if (graph[u][v] && mstSet[v] == false && graph[u][v] < key[v]) {
39                 parent[v] = u, key[v] = graph[u][v];
40             }
41     }
42     printMST(parent);
43 }
44 int main() {
45     printf("Enter the number of vertices: ");
46     scanf("%d", &n);
47     printf("Enter the adjacency matrix:\n");
48     for (int i = 0; i < n; i++)
49         for (int j = 0; j < n; j++)
50             scanf("%d", &graph[i][j]);
51     primMST();
52     return 0;
53 }
```

```
0 - 3 6
1 - 4 5
Total cost of Minimum Spanning Tree: 16

=== Code Execution Successful ===
```

3. Write a C program for KRUSKAL'S Algorithm.

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #define MAX 100
4 typedef struct {
5     int u, v, w;
6 } Edge;
7 typedef struct {
8     Edge edges[MAX];
9     int n;
10 } Edgelist;
11 Edgelist elist;
12 int parent[MAX];
13 Edgelist spanlist;
14 int find(int v) {
15     if (parent[v] == v)
16         return v;
17     return find(parent[v]);
18 }
19 void union_set(int u, int v) {
20     parent[u] = v;
21 }
22 void kruskal(int n) {
23     int i, u, v;
24     for (i = 0; i < n; i++) {
25         parent[i] = i;
26     }
27     spanlist.n = 0;
28     for (i = 0; i < elist.n; i++) {
29         u = find(elist.edges[i].u);
30         v = find(elist.edges[i].v);
31         if (u != v) {
32             spanlist.edges[spanlist.n] = elist.edges[i];
33             spanlist.n = spanlist.n + 1;
34             union_set(u, v);

```

```

/tmp/ipuEUSXCx7.o
Enter the number of vertices: 4
Enter the number of edges: 5
Enter edge 1 (u, v, weight): 0 1 10
Enter edge 2 (u, v, weight): 0 2 6
Enter edge 3 (u, v, weight): 0 3 5
Enter edge 4 (u, v, weight): 1 3 15
Enter edge 5 (u, v, weight): 2 3 4
Edges in the Minimum Spanning Tree:
2 - 3 : 4
0 - 3 : 5
0 - 1 : 10
Total cost of Minimum Spanning Tree: 19

```

=== Code Execution Successful ===

```

33     spanlist.n = spanlist.n + 1;
34     union_set(u, v);
35 }
36 }
37 }
38 void sort() {
39     int i, j;
40     Edge temp;
41     for (i = 1; i < elist.n; i++)
42         for (j = 0; j < elist.n - i; j++)
43             if (elist.edges[j].w > elist.edges[j + 1].w) {
44                 temp = elist.edges[j];
45                 elist.edges[j] = elist.edges[j + 1];
46                 elist.edges[j + 1] = temp;
47             }
48 }
49 void print() {
50     int i, cost = 0;
51
52     printf("Edges in the Minimum Spanning Tree:\n");
53     for (i = 0; i < spanlist.n; i++) {
54         printf("%d - %d : %d\n", spanlist.edges[i].u, spanlist.edges[i].v,
55             spanlist.edges[i].w);
56         cost += spanlist.edges[i].w;
57     }
58     printf("Total cost of Minimum Spanning Tree: %d\n", cost);
59 }
60 int main() {
61     int n, e, i;
62     printf("Enter the number of vertices: ");
63     scanf("%d", &n);
64     printf("Enter the number of edges: ");
65     scanf("%d", &e);
66     elist.n = e;

```

```

/tmp/ipuEUSXCx7.o
Enter the number of vertices: 4
Enter the number of edges: 5
Enter edge 1 (u, v, weight): 0 1 10
Enter edge 2 (u, v, weight): 0 2 6
Enter edge 3 (u, v, weight): 0 3 5
Enter edge 4 (u, v, weight): 1 3 15
Enter edge 5 (u, v, weight): 2 3 4
Edges in the Minimum Spanning Tree:
2 - 3 : 4
0 - 3 : 5
0 - 1 : 10
Total cost of Minimum Spanning Tree: 19

```

=== Code Execution Successful ===

```

61     printf("Enter the number of vertices: ");
62     scanf("%d", &n);
63     printf("Enter the number of edges: ");
64     scanf("%d", &e);
65     elist.n = e;
66     for (i = 0; i < e; i++) {
67         printf("Enter edge %d (u, v, weight): ", i + 1);
68         scanf("%d%d%d", &elist.edges[i].u, &elist.edges[i].v, &elist.edges[i].w);
69     }
70     sort();
71     kruskal(n);
72     print();
73     return 0;
74 }

```

