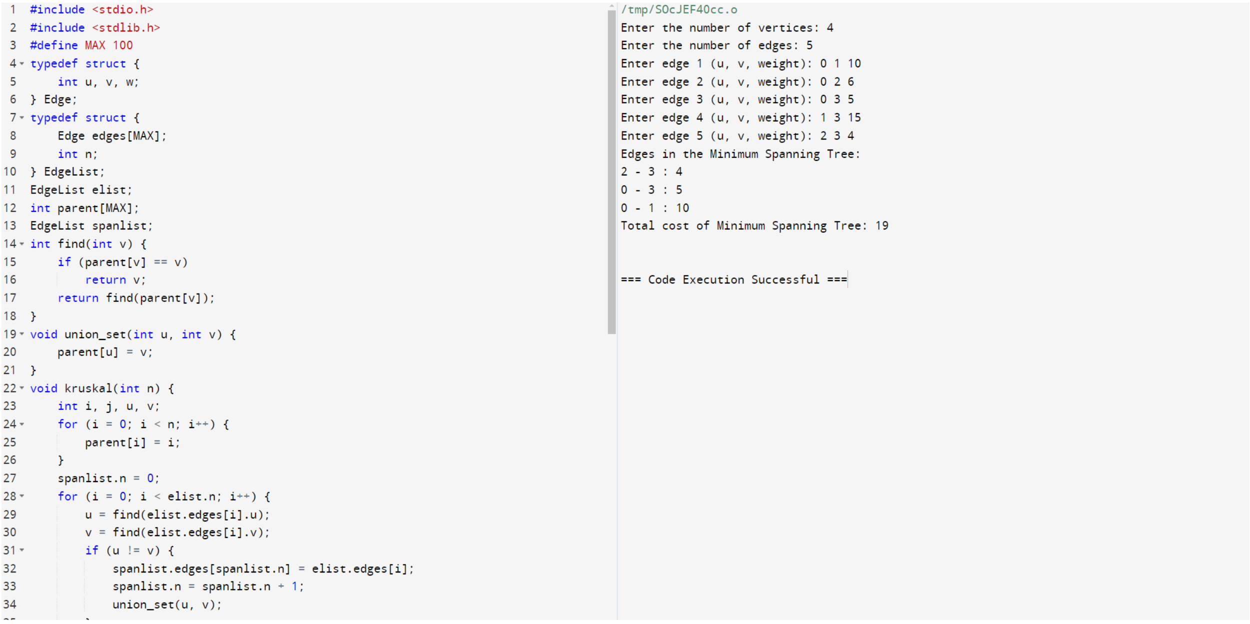
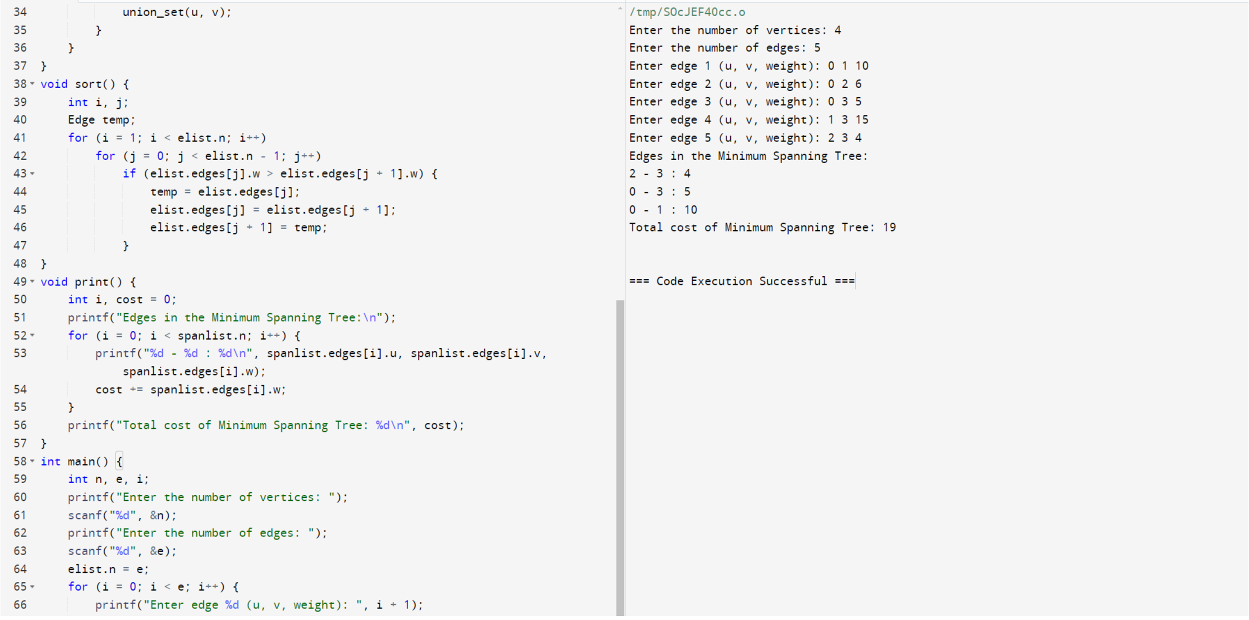
1. Write a c program for Minimum Spanning Tree.



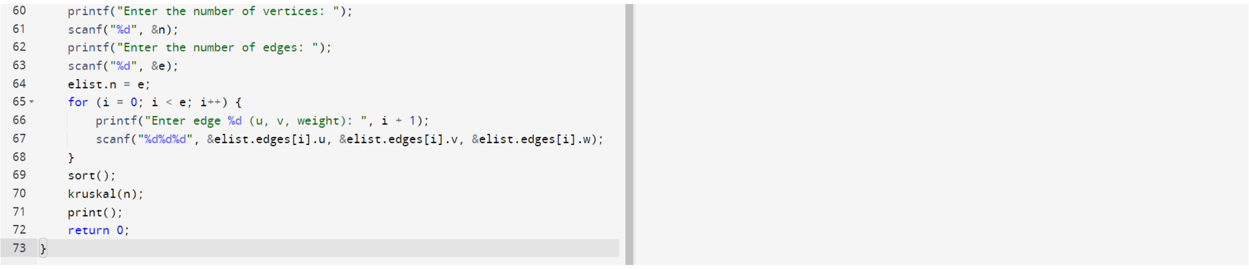
```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 100
typedef struct {
    int u, v, w;
} Edge;
typedef struct {
    Edge edges[MAX];
    int n;
} EdgeList;
EdgeList elist;
int parent[MAX];
EdgeList spanlist;
int find(int v) {
    if (parent[v] == v)
        return v;
    return find(parent[v]);
}
void union_set(int u, int v) {
    parent[u] = v;
}
void kruskal(int n) {
    int i, j, u, v;
    for (i = 0; i < n; i++) {
        parent[i] = i;
    }
    spanlist.n = 0;
    for (i = 0; i < elist.n; i++) {
        u = find(elist.edges[i].u);
        v = find(elist.edges[i].v);
        if (u != v) {
            spanlist.edges[spanlist.n] = elist.edges[i];
            spanlist.n = spanlist.n + 1;
            union_set(u, v);
```

```
/tmp/SOcJEF40cc.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 ===
```



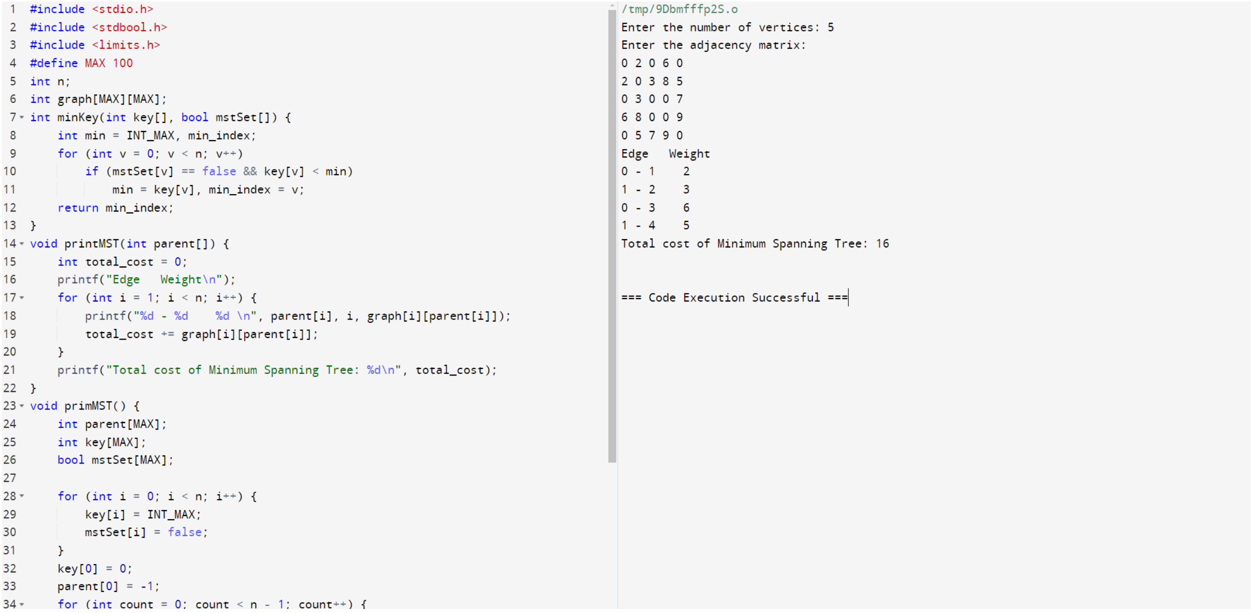
```c
            union_set(u, v);
        }
    }
}
void sort() {
    int i, j;
    Edge temp;
    for (i = 1; i < elist.n; i++)
        for (j = 0; j < elist.n - 1; j++)
            if (elist.edges[j].w > elist.edges[j + 1].w) {
                temp = elist.edges[j];
                elist.edges[j] = elist.edges[j + 1];
                elist.edges[j + 1] = temp;
            }
}
void print() {
    int i, cost = 0;
    printf("Edges in the Minimum Spanning Tree:\n");
    for (i = 0; i < spanlist.n; i++) {
        printf("%d - %d : %d\n", spanlist.edges[i].u, spanlist.edges[i].v,
            spanlist.edges[i].w);
        cost += spanlist.edges[i].w;
    }
    printf("Total cost of Minimum Spanning Tree: %d\n", cost);
}
int main() {
    int n, e, i;
    printf("Enter the number of vertices: ");
    scanf("%d", &n);
    printf("Enter the number of edges: ");
    scanf("%d", &e);
    elist.n = e;
    for (i = 0; i < e; i++) {
        printf("Enter edge %d (u, v, weight): ", i + 1);
```

```
/tmp/SOcJEF40cc.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 ===
```



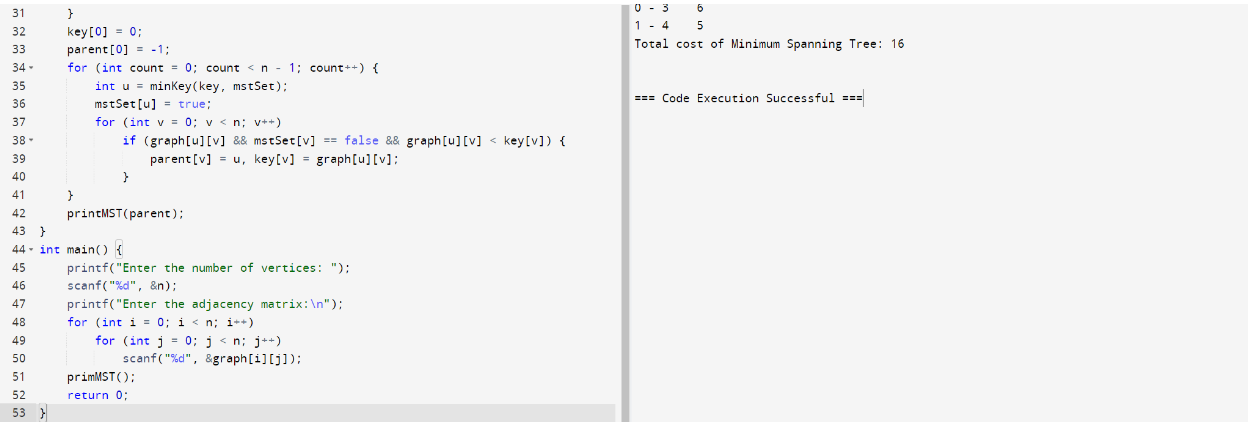
```c
    printf("Enter the number of vertices: ");
    scanf("%d", &n);
    printf("Enter the number of edges: ");
    scanf("%d", &e);
    elist.n = e;
    for (i = 0; i < e; i++) {
        printf("Enter edge %d (u, v, weight): ", i + 1);
        scanf("%d%d%d", &elist.edges[i].u, &elist.edges[i].v, &elist.edges[i].w);
    }
    sort();
    kruskal(n);
    print();
    return 0;
}
```

2. Write a C program for Prims Algorithm.



```c
#include <stdio.h>
#include <stdbool.h>
#include <limits.h>
#define MAX 100
int n;
int graph[MAX][MAX];
int minKey(int key[], bool mstSet[]) {
    int min = INT_MAX, min_index;
    for (int v = 0; v < n; v++)
        if (mstSet[v] == false && key[v] < min)
            min = key[v], min_index = v;
    return min_index;
}
void printMST(int parent[]) {
    int total_cost = 0;
    printf("Edge   Weight\n");
    for (int i = 1; i < n; i++) {
        printf("%d - %d    %d \n", parent[i], i, graph[i][parent[i]]);
        total_cost += graph[i][parent[i]];
    }
    printf("Total cost of Minimum Spanning Tree: %d\n", total_cost);
}
void primMST() {
    int parent[MAX];
    int key[MAX];
    bool mstSet[MAX];

    for (int i = 0; i < n; i++) {
        key[i] = INT_MAX;
        mstSet[i] = false;
    }
    key[0] = 0;
    parent[0] = -1;
    for (int count = 0; count < n - 1; count++) {
```

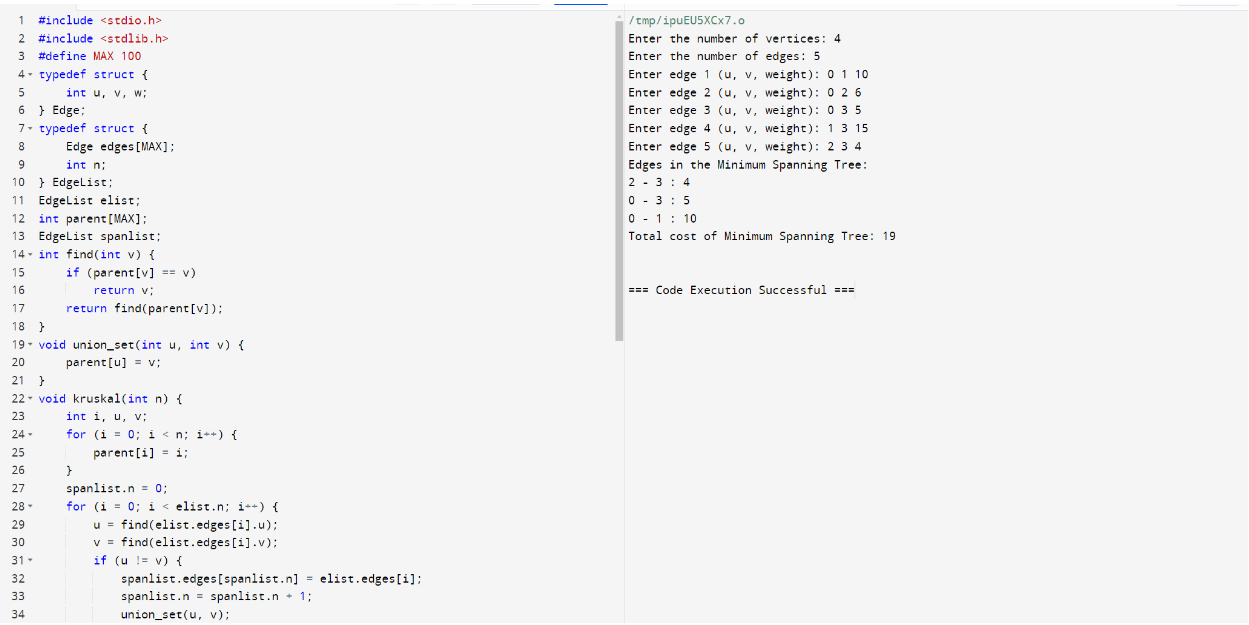
```
/tmp/9Dbmfffp2S.o
Enter the number of vertices: 5
Enter the adjacency matrix:
0 2 0 6 0
2 0 3 8 5
0 3 0 0 7
6 8 0 0 9
0 5 7 9 0
Edge   Weight
0 - 1    2
1 - 2    3
0 - 3    6
1 - 4    5
Total cost of Minimum Spanning Tree: 16


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



```c
    }
    key[0] = 0;
    parent[0] = -1;
    for (int count = 0; count < n - 1; count++) {
        int u = minKey(key, mstSet);
        mstSet[u] = true;
        for (int v = 0; v < n; v++)
            if (graph[u][v] && mstSet[v] == false && graph[u][v] < key[v]) {
                parent[v] = u, key[v] = graph[u][v];
            }
    }
    printMST(parent);
}
int main() {
    printf("Enter the number of vertices: ");
    scanf("%d", &n);
    printf("Enter the adjacency matrix:\n");
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            scanf("%d", &graph[i][j]);
    primMST();
    return 0;
}
```

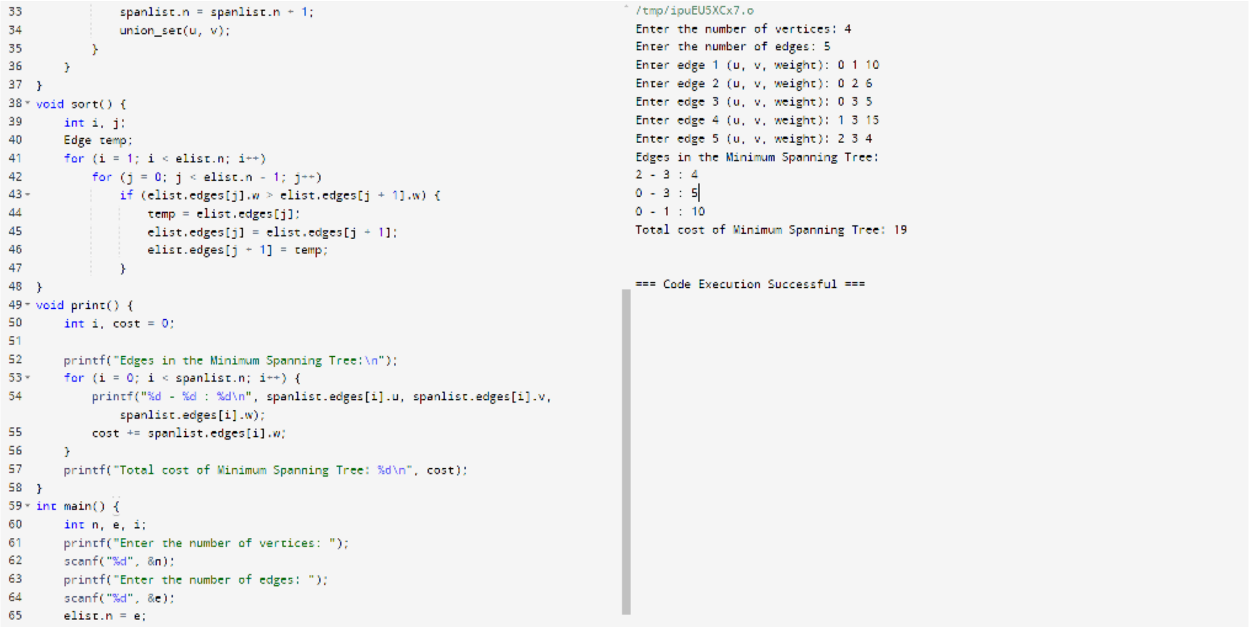
```
0 - 3    6
1 - 4    5
Total cost of Minimum Spanning Tree: 16


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

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



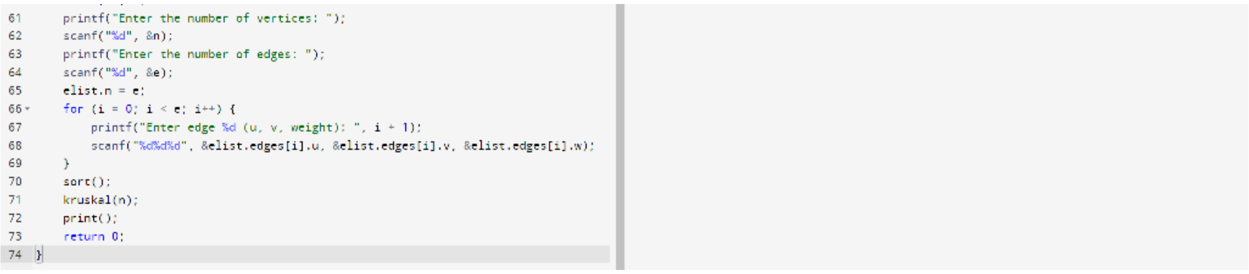
```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 100
typedef struct {
    int u, v, w;
} Edge;
typedef struct {
    Edge edges[MAX];
    int n;
} EdgeList;
EdgeList elist;
int parent[MAX];
EdgeList spanlist;
int find(int v) {
    if (parent[v] == v)
        return v;
    return find(parent[v]);
}
void union_set(int u, int v) {
    parent[u] = v;
}
void kruskal(int n) {
    int i, u, v;
    for (i = 0; i < n; i++) {
        parent[i] = i;
    }
    spanlist.n = 0;
    for (i = 0; i < elist.n; i++) {
        u = find(elist.edges[i].u);
        v = find(elist.edges[i].v);
        if (u != v) {
            spanlist.edges[spanlist.n] = elist.edges[i];
            spanlist.n = spanlist.n + 1;
            union_set(u, v);
```

```
/tmp/ipuEU5XCx7.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 ===
```



```c
            spanlist.n = spanlist.n + 1;
            union_set(u, v);
        }
    }
}
void sort() {
    int i, j;
    Edge temp;
    for (i = 1; i < elist.n; i++)
        for (j = 0; j < elist.n - 1; j++)
            if (elist.edges[j].w > elist.edges[j + 1].w) {
                temp = elist.edges[j];
                elist.edges[j] = elist.edges[j + 1];
                elist.edges[j + 1] = temp;
            }
}
void print() {
    int i, cost = 0;

    printf("Edges in the Minimum Spanning Tree:\n");
    for (i = 0; i < spanlist.n; i++) {
        printf("%d - %d : %d\n", spanlist.edges[i].u, spanlist.edges[i].v,
            spanlist.edges[i].w);
        cost += spanlist.edges[i].w;
    }
    printf("Total cost of Minimum Spanning Tree: %d\n", cost);
}
int main() {
    int n, e, i;
    printf("Enter the number of vertices: ");
    scanf("%d", &n);
    printf("Enter the number of edges: ");
    scanf("%d", &e);
    elist.n = e;
```

```
/tmp/ipuEU5XCx7.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 ===
```



```c
    printf("Enter the number of vertices: ");
    scanf("%d", &n);
    printf("Enter the number of edges: ");
    scanf("%d", &e);
    elist.n = e;
    for (i = 0; i < e; i++) {
        printf("Enter edge %d (u, v, weight): ", i + 1);
        scanf("%d%d%d", &elist.edges[i].u, &elist.edges[i].v, &elist.edges[i].w);
    }
    sort();
    kruskal(n);
    print();
    return 0;
}
```