Exp_09\minSpaningTree.c

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int src, dest, weight;
} Edge;

typedef struct {
    int parent, rank;
} Subset;

int find(Subset subsets[], int i) {
    if (subsets[i].parent != i)
        subsets[i].parent = find(subsets, subsets[i].parent);
    return subsets[i].parent;
}

void unionSets(Subset subsets[], int x, int y) {
    int xroot = find(subsets, x);
    int yroot = find(subsets, y);

    if (subsets[xroot].rank < subsets[yroot].rank)
        subsets[xroot].parent = yroot;
    else if (subsets[xroot].rank > subsets[yroot].rank)
        subsets[yroot].parent = xroot;
    else {
        subsets[yroot].parent = xroot;
        subsets[xroot].rank++;
    }
}

int compareEdges(const void* a, const void* b) {
    return ((Edge*)a)->weight - ((Edge*)b)->weight;
}

void kruskalMST(Edge edges[], int V, int E) {
    qsort(edges, E, sizeof(Edge), compareEdges);

    Subset* subsets = (Subset*)malloc(V * sizeof(Subset));
    for (int i = 0; i < V; i++) {
        subsets[i].parent = i;
        subsets[i].rank = 0;
    }

    Edge result[V];
    int e = 0, i = 0;

    printf("Edges in MST:\n");
    while (e < V - 1 && i < E) {
        Edge next_edge = edges[i++];
        int x = find(subsets, next_edge.src);
```

```c
        int y = find(subsets, next_edge.dest);

        if (x != y) {
            result[e++] = next_edge;
            printf("%d -- %d == %d\n", next_edge.src, next_edge.dest,
next_edge.weight);
            unionSets(subsets, x, y);
        }
    }

    int totalWeight = 0;
    for (int i = 0; i < e; i++)
        totalWeight += result[i].weight;
    printf("Total weight of MST: %d\n", totalWeight);

    free(subsets);
}

int main() {
    int V, E;
    printf("Enter number of vertices: ");
    scanf("%d", &V);
    printf("Enter number of edges: ");
    scanf("%d", &E);

    Edge* edges = (Edge*)malloc(E * sizeof(Edge));
    printf("Enter edges (src dest weight):\n");
    for (int i = 0; i < E; i++) {
        scanf("%d %d %d", &edges[i].src, &edges[i].dest, &edges[i].weight);
    }

    kruskalMST(edges, V, E);

    free(edges);
    return 0;
}
```