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

#define V 5


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


struct subset {
    int parent;
    int rank;
};


int find(struct subset subsets[], int i);
void Union(struct subset subsets[], int x, int y);
int compare(const void* a, const void* b);
void KruskalMST(struct Edge edges[]);


int find(struct subset subsets[], int i) {
   i
    if (subsets[i].parent != i)
        subsets[i].parent = find(subsets, subsets[i].parent);

    return subsets[i].parent;
}


void Union(struct 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 compare(const void* a, const void* b) {
    struct Edge* edge1 = (struct Edge*)a;
    struct Edge* edge2 = (struct Edge*)b;
    return edge1->weight - edge2->weight;
}


void KruskalMST(struct Edge edges[]) {
    struct Edge result[V];
    int e = 0;


    qsort(edges, V, sizeof(edges[0]), compare);


    struct subset* subsets = (struct subset*)malloc(V * sizeof(struct subset));


    for (int v = 0; v < V; ++v) {
        subsets[v].parent = v;
        subsets[v].rank = 0;
    }


    while (e < V - 1 && edges[e].weight != 0) {
```

```c
        struct Edge next_edge = edges[e++];

        int x = find(subsets, next_edge.src);
        int y = find(subsets, next_edge.dest);

        if (x != y) {
            result[e - 1] = next_edge;
            Union(subsets, x, y);
        }
    }

    printf("Edges in the Minimum Spanning Tree:\n");
    for (int i = 0; i < e - 1; ++i)
        printf("%d - %d: %d\n", result[i].src, result[i].dest, result[i].weight);

    free(subsets);
}


int main() {

    struct Edge edges[] = {
        {0, 1, 2},
        {0, 3, 6},
        {1, 2, 3},
        {1, 3, 8},
        {1, 4, 5},
        {2, 4, 7},
        {3, 4, 9}
    };


    KruskalMST(edges);

    return 0;
}
```