Q. You have a business with several offices. You have to lease phone lines to connect them with each other and the phone company charges different amounts of money to connect different pairs of cities. You want a set of lines that connects all your offices with a minimum total cost. Solve the problem by suggesting an appropriate data structure.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 100
struct Edge {
    int u, v, weight;
};
int parent[MAX], rank[MAX];
void initializeSet(int n) {
    for (int i = 0; i < n; i++) {
        parent[i] = i;
        rank[i] = 0;
    }
}

int find(int i) {
    if (parent[i] != i) {
        parent[i] = find(parent[i]);
    }
    return parent[i];
}

void unionSets(int u, int v) {
    int rootU = find(u);
    int rootV = find(v);

    if (rootU != rootV) {
        if (rank[rootU] > rank[rootV]) {
```

```
            parent[rootV] = rootU;
        } else if (rank[rootU] < rank[rootV]) {
            parent[rootU] = rootV;
        } else {
            parent[rootV] = rootU;
            rank[rootU]++;
        }
    }
}

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

void kruskal(struct Edge edges[], int e, int n) {
    int mstWeight = 0;
    int edgesIncluded = 0;

    qsort(edges, e, sizeof(struct Edge), compareEdges);
    initializeSet(n);

    printf("Edges in the MST:\n");
    for (int i = 0; i < e; i++) {
        int u = edges[i].u;
        int v = edges[i].v;
        int weight = edges[i].weight;

        if (find(u) != find(v)) {
            printf("Office %d - Office %d with cost %d\n", u, v, weight);
            unionSets(u, v);
            mstWeight += weight;
```

```
            edgesIncluded++;

            if (edgesIncluded == n - 1) {
                break;
            }
        }
    }

    if (edgesIncluded == n - 1) {
        printf("\nMinimum cost to connect all offices: %d\n", mstWeight);
    } else {
        printf("There is no way to connect all offices.\n");
    }
}

int main() {
    int n, e, choice;

    printf("Enter number of offices (cities): ");
    scanf("%d", &n);

    printf("Enter number of possible connections (edges): ");
    scanf("%d", &e);

    struct Edge edges[e];

    for (int i = 0; i < e; i++) {
        printf("\nEnter the offices connected by edge %d: ", i + 1);
        scanf("%d %d", &edges[i].u, &edges[i].v);
        printf("Enter the cost of connecting office %d and office %d: ", edges[i].u, edges[i].v);
        scanf("%d", &edges[i].weight);
```

```
    }

    do {
        printf("\nChoose an option:\n");
        printf("1. Calculate Minimum Spanning Tree (MST) using Kruskal's algorithm\n");
        printf("2. Exit\n");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                kruskal(edges, e, n);
                break;
            case 2:
                printf("Exiting program...\n");
                break;
            default:
                printf("Invalid choice, please try again.\n");
        }
    } while (choice != 2);
  return 0;
}
```

Output Clear

```
Enter number of offices (cities): 4
Enter number of possible connections (edges): 2

Enter the offices connected by edge 1: 6
4
Enter the cost of connecting office 6 and office 4: 2

Enter the offices connected by edge 2: 8 5
Enter the cost of connecting office 8 and office 5: 13

Choose an option:
1. Calculate Minimum Spanning Tree (MST) using Kruskal's algorithm
2. Exit
1
Edges in the MST:
There is no way to connect all offices.

Choose an option:
1. Calculate Minimum Spanning Tree (MST) using Kruskal's algorithm
2. Exit

=== Session Ended. Please Run the code again ===
```