```c
/*
1. Write a C program to input a weighted undirected graph and generate its Minimum
Spanning Tree (MST) using Kruskal's Algorithm. The program must print the edges in
MST and the total MST weight.
*/

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

// Structure to represent an edge
struct Edge
{
    int src, dest, weight;
};

// Structure to represent a subset for Union-Find
struct Subset
{
    int parent;
    int rank;
};

// Find function for Union-Find
int find(struct Subset subsets[], int i)
{
    if (subsets[i].parent != i)
        subsets[i].parent = find(subsets, subsets[i].parent);
    return subsets[i].parent;
}

// Union function for Union-Find
```

```c
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++;
    }
}

// Compare function for qsort
int compareEdges(const void *a, const void *b)
{
    return ((struct Edge *)a)->weight - ((struct Edge *)b)->weight;
}

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

    struct Edge edges[E];
    printf("Enter each edge as: src dest weight\n");
    for (int i = 0; i < E; i++)
    {
```

```
        scanf("%d %d %d", &edges[i].src, &edges[i].dest,
&edges[i].weight);
    }

    // Sort edges by weight
    qsort(edges, E, sizeof(edges[0]), compareEdges);

    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;
    }

    int mstWeight = 0;
    printf("\nEdges in MST (Kruskal):\n");
    for (int i = 0, e = 0; e < V - 1 && i < E; i++)
    {
        struct Edge nextEdge = edges[i];
        int x = find(subsets, nextEdge.src);
        int y = find(subsets, nextEdge.dest);

        if (x != y)
        {
            printf("%d -- %d == %d\n", nextEdge.src, nextEdge.dest,
nextEdge.weight);
            mstWeight += nextEdge.weight;
            Union(subsets, x, y);
            e++;
        }
    }
```

```
    printf("Total MST weight = %d\n", mstWeight);
    return 0;
}
```

Question 2:

```
/*
2. Write a C program that reads an adjacency matrix of a connected
weighted graph and
generates its MST using Prim's Algorithm. The program must print the
edges selected at
each step and the total MST weight.
*/

#include <stdio.h>
#include <limits.h>

#define MAX 100

int minKey(int key[], int mstSet[], int V)
{
    int min = INT_MAX, min_index;
    for (int v = 0; v < V; v++)
    {
        if (mstSet[v] == 0 && key[v] < min)
        {
            min = key[v];
            min_index = v;
        }
    }
    return min_index;
}
```

```c
void primMST(int graph[MAX][MAX], int V)
{
    int parent[V]; // Store MST
    int key[V];    // Key values
    int mstSet[V]; // Included in MST?

    for (int i = 0; i < V; i++)
    {
        key[i] = INT_MAX;
        mstSet[i] = 0;
    }

    key[0] = 0;     // Start from vertex 0
    parent[0] = -1; // First node is root

    for (int count = 0; count < V - 1; count++)
    {
        int u = minKey(key, mstSet, V);
        mstSet[u] = 1;

        for (int v = 0; v < V; v++)
        {
            if (graph[u][v] && mstSet[v] == 0 && graph[u][v] < key[v])
            {
                parent[v] = u;
                key[v] = graph[u][v];
            }
        }
    }

    int totalWeight = 0;
    printf("\nEdges in MST (Prim):\n");
```

```
    for (int i = 1; i < V; i++)
    {
        printf("%d -- %d == %d\n", parent[i], i, graph[i][parent[i]]);
        totalWeight += graph[i][parent[i]];
    }
    printf("Total MST weight = %d\n", totalWeight);
}

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

    int graph[MAX][MAX];
    printf("Enter adjacency matrix (%d x %d):\n", V, V);
    for (int i = 0; i < V; i++)
    {
        for (int j = 0; j < V; j++)
        {
            scanf("%d", &graph[i][j]);
        }
    }

    primMST(graph, V);
    return 0;
}
```