1. **Implementation of Minimum Spanning Tree using Kruskal Algorithm.**

**PROGRAM:**

#include <stdio.h>

#include <stdlib.h>

#define MAX 100

struct Edge {

int src, dest, weight;

};

struct Graph {

int V, E;

struct Edge\* edge;

};

struct subset {

int parent;

int rank;

};

struct Graph\* createGraph(int V, int E) {

struct Graph\* graph = (struct Graph\*) malloc(sizeof(struct Graph));

graph->V = V;

graph->E = E;

graph->edge = (struct Edge\*) malloc(graph->E \* sizeof(struct Edge));

return graph;

}

int find(struct subset subsets[], int 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 rootX = find(subsets, x);

int rootY = find(subsets, y);

if (subsets[rootX].rank < subsets[rootY].rank)

subsets[rootX].parent = rootY;

else if (subsets[rootX].rank > subsets[rootY].rank)

subsets[rootY].parent = rootX;

else {

subsets[rootY].parent = rootX;

subsets[rootX].rank++;

}

}

int compareEdges(const void\* a, const void\* b) {

struct Edge\* a1 = (struct Edge\*) a;

struct Edge\* b1 = (struct Edge\*) b;

return a1->weight > b1->weight;

}

// Function to construct MST using Kruskal's algorithm

void KruskalMST(struct Graph\* graph) {

int V = graph->V;

struct Edge result[MAX];

int e = 0; // Index variable for result[]

int i = 0; // Index variable for sorted edges

// Step 1: Sort all the edges in non-decreasing order of their weight

qsort(graph->edge, graph->E, sizeof(graph->edge[0]), compareEdges);

// Allocate memory for creating V subsets

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

// Create V subsets with single elements

for (int v = 0; v < V; ++v) {

subsets[v].parent = v;

subsets[v].rank = 0;

}

// Number of edges to be taken is equal to V-1

while (e < V - 1 && i < graph->E) {

// Step 2: Pick the smallest edge and increment the index for next iteration

struct Edge next\_edge = graph->edge[i++];

int x = find(subsets, next\_edge.src);

int y = find(subsets, next\_edge.dest);

// If including this edge doesn't cause cycle, include it in result

// and increment the index of result for next edge

if (x != y) {

result[e++] = next\_edge;

Union(subsets, x, y);

}

}

// Print the constructed MST

printf("Following are the edges in the constructed MST:\n");

for (i = 0; i < e; ++i)

printf("%d -- %d == %d\n", result[i].src, result[i].dest, result[i].weight);

free(subsets);

}

int main() {

int V, E;

printf("Enter the number of vertices and edges: ");

scanf("%d %d", &V, &E);

struct Graph\* graph = createGraph(V, E);

printf("Enter the edges (source destination weight):\n");

for (int i = 0; i < E; i++) {

scanf("%d %d %d", &graph->edge[i].src, &graph->edge[i].dest, &graph->edge[i].weight);

}

KruskalMST(graph);

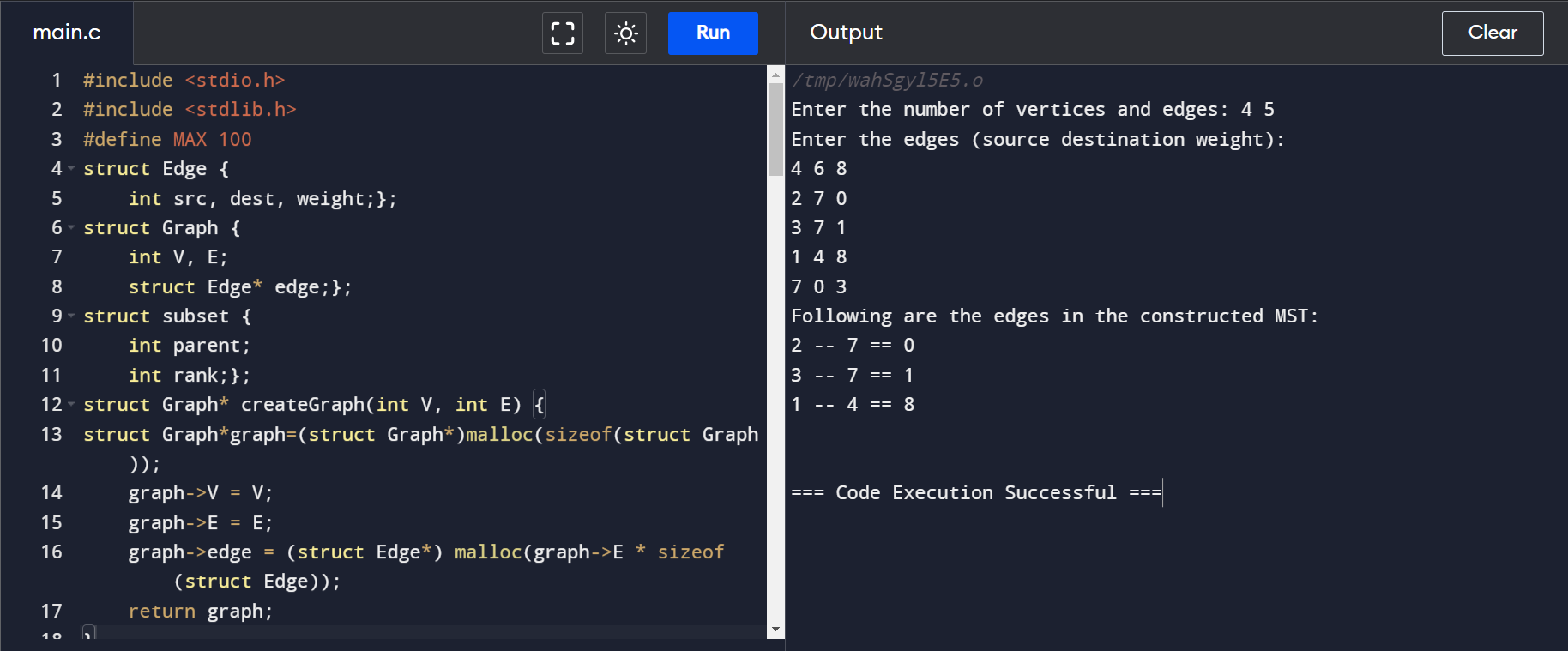
free(graph->edge);

free(graph);

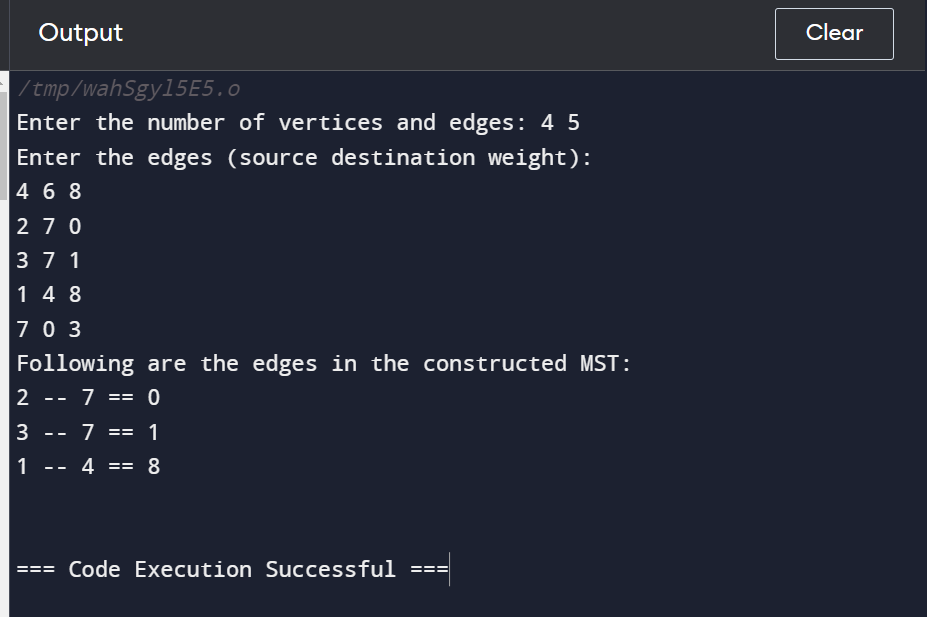
return 0;

}

**INPUT:**

****

**OUTPUT:**

****