#include <stdio.h>

#include <stdlib.h>

#define MAX 100

// 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;

};

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 unionSets(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;

}

void kruskalMST(struct Edge edges[], int V, int E) {

struct Edge result[MAX]; // Store MST edges

struct Subset subsets[MAX];

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

int i = 0; // Index for sorted edges

// Sort edges by weight

qsort(edges, E, sizeof(struct Edge), compareEdges);

// Initialize subsets

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

subsets[v].parent = v;

subsets[v].rank = 0;

}

// Pick edges one by one

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

struct Edge next = edges[i++];

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

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

// If including this edge doesn't cause a cycle

if (x != y) {

result[e++] = next;

unionSets(subsets, x, y);

}

}

// Print MST

int totalCost = 0;

printf("Edges in Minimum Spanning Tree:\n");

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

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

totalCost += result[i].weight;

}

printf("Total cost of MST: %d\n", totalCost);

}

int main() {

int V, E;

struct Edge edges[MAX];

// Input: number of vertices and edges

printf("Enter number of vertices: ");

scanf("%d", &V);

printf("Enter number of edges: ");

scanf("%d", &E);

// Input: edges

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);

}

// Run Kruskal's Algorithm

kruskalMST(edges, V, E);

return 0;

}