#include <stdio.h>

#include <stdlib.h>

#define MAX 100

typedef struct {

int u, v, w;

} Edge;

int parent[MAX];

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

return ((Edge \*)a)->w - ((Edge \*)b)->w;

}

int find(int i) {

if (parent[i] == -1)

return i;

return find(parent[i]);

}

void union\_set(int x, int y) {

parent[x] = y;

}

void kruskal(Edge edges[], int n, int m) {

int total\_cost = 0;

int edge\_count = 0;

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

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

int u = find(edges[i].u);

int v = find(edges[i].v);

if (u != v) {

union\_set(u, v);

total\_cost += edges[i].w;

edge\_count++;

printf("Edge (%d, %d) with weight %d included in MST\n", edges[i].u, edges[i].v, edges[i].w);

}

if (edge\_count == n - 1)

break;

}

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

}

int main() {

int n, m;

Edge edges[MAX];

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

scanf("%d", &n);

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

scanf("%d", &m);

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

parent[i] = -1;

}

printf("Enter the edges (u, v, weight):\n");

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

scanf("%d %d %d", &edges[i].u, &edges[i].v, &edges[i].w);

}

kruskal(edges, n, m);

return 0;

}

Enter the number of vertices: 6

Enter the number of edges: 8

Enter the edges (u, v, weight):

0 1 4

0 2 4

1 2 2

2 3 3

2 4 4

2 5 2

4 5 3

3 4 3

Edge (1, 2) with weight 2 included in MST

Edge (2, 5) with weight 2 included in MST

Edge (2, 3) with weight 3 included in MST

Edge (4, 5) with weight 3 included in MST

Edge (0, 2) with weight 4 included in MST

Total cost of MST: 14