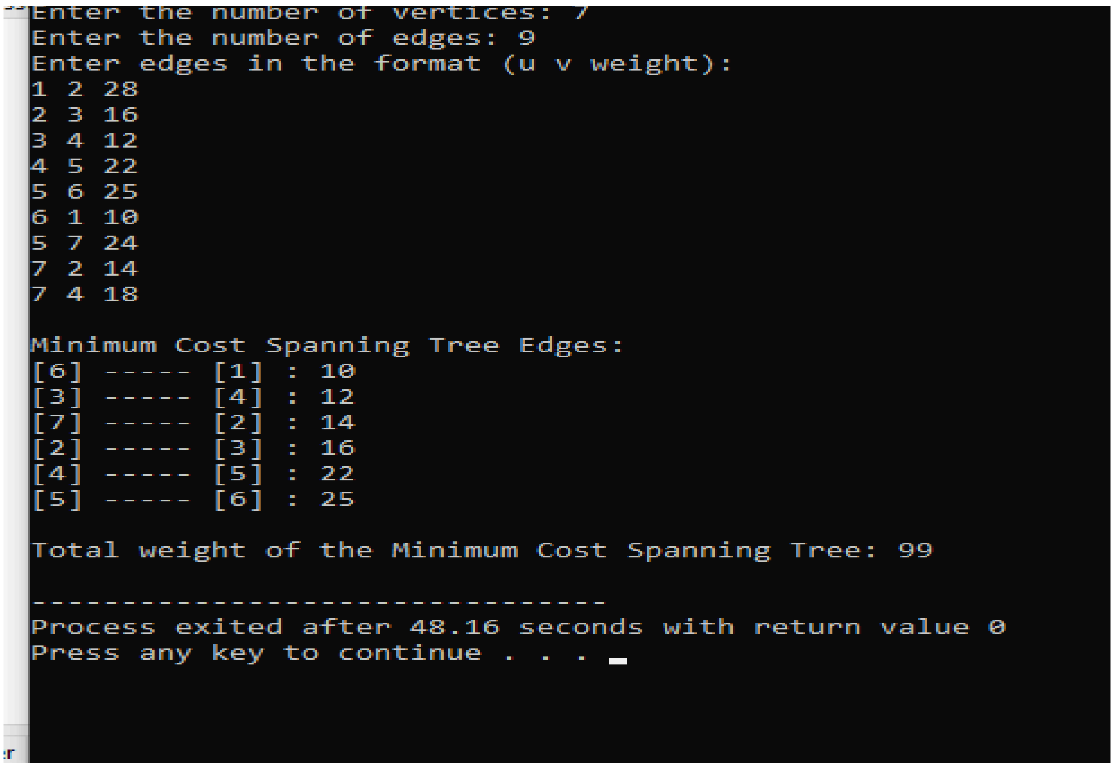
Assignment 7

```
#include <iostream>

using namespace std;

struct Edge {
    int u, v, weight;
};

class DisjointSet {
public:
    DisjointSet(int n) {
        for (int i = 0; i < n; ++i) {
            parent[i] = i;
        }
    }

    int find(int u) {
        if (parent[u] != u) {
            parent[u] = find(parent[u]);  // Path compression
        }
        return parent[u];
    }

    void unionSets(int u, int v) {
        int rootU = find(u);
        int rootV = find(v);
        if (rootU != rootV) {
            parent[rootV] = rootU;  // Union
        }
    }

private:
    int parent[100];  // Assuming a maximum of 100 vertices
};

// Function to sort edges using bubble sort
void bubbleSort(Edge edges[], int E) {
    for (int i = 0; i < E - 1; i++) {
        for (int j = 0; j < E - i - 1; j++) {
            if (edges[j].weight > edges[j + 1].weight) {
                // Swap edges
                Edge temp = edges[j];
                edges[j] = edges[j + 1];
                edges[j + 1] = temp;
            }
        }
    }
}

int main() {
    int V, E;
    cout << "Enter the number of vertices: ";
    cin >> V;
    cout << "Enter the number of edges: ";
    cin >> E;

    Edge edges[100];  // Assuming a maximum of 100 edges

    cout << "Enter edges in the format (u v weight):\n";
    for (int i = 0; i < E; ++i) {
        cin >> edges[i].u >> edges[i].v >> edges[i].weight;
        // Convert to zero-based indexing
        edges[i].u--; // Decrementing to use zero-based index
        edges[i].v--; // Decrementing to use zero-based index
    }

    // Sort edges based on weight using bubble sort
    bubbleSort(edges, E);

    DisjointSet ds(V);
    int totalWeight = 0;

    cout << "\nMinimum Cost Spanning Tree Edges:\n";
    for (int i = 0; i < E; ++i) {
        if (ds.find(edges[i].u) != ds.find(edges[i].v)) {
            ds.unionSets(edges[i].u, edges[i].v);
            totalWeight += edges[i].weight;
            cout << "[" << edges[i].u + 1 << "] ----- [" << edges[i].v + 1 << "] : " << edges[i].weight
<< endl; // Convert back to 1-based for output
        }
    }

    cout << "\nTotal weight of the Minimum Cost Spanning Tree: " << totalWeight << endl;

    return 0;
}
```

Output:



```
Enter the number of vertices: 7
Enter the number of edges: 9
Enter edges in the format (u v weight):
1 2 28
2 3 16
3 4 12
4 5 22
5 6 25
6 1 10
5 7 24
7 2 14
7 4 18

Minimum Cost Spanning Tree Edges:
[6] ----- [1] : 10
[3] ----- [4] : 12
[7] ----- [2] : 14
[2] ----- [3] : 16
[4] ----- [5] : 22
[5] ----- [6] : 25

Total weight of the Minimum Cost Spanning Tree: 99

--------------------------------
Process exited after 48.16 seconds with return value 0
Press any key to continue . . .
```