Assignment No. 8

Name – Manish Namdev Barage

PRN – 22520007

Batch – T7

Problem Statement – To minimum cost spanning tree of a given undirected graph using Kruskal's Algorithm

Code –

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <utility>

using namespace std;

class Graph {
private:
    int numVertices;
    vector<pair<int, pair<int, int>>> edges;

public:
    Graph(int vertices) {
        numVertices = vertices;
    }

    void addEdge(int src, int dest, int weight) {
        edges.push_back({weight, {src, dest}});
    }

    int findParent(int vertex, vector<int>& parent) {
        if (parent[vertex] != vertex) {
            parent[vertex] = findParent(parent[vertex], parent);
        }
        return parent[vertex];
    }

    void unionSets(int u, int v, vector<int>& parent, vector<int>&
rank) {
        int rootU = findParent(u, parent);
        int rootV = findParent(v, parent);

        if (rootU != rootV) {
            if (rank[rootU] > rank[rootV]) {
                parent[rootV] = rootU;
            } else if (rank[rootU] < rank[rootV]) {
                parent[rootU] = rootV;
            } else {
                parent[rootV] = rootU;
                rank[rootU]++;
```

```
            }
        }
    }

    void kruskalMST() {
        sort(edges.begin(), edges.end());

        vector<int> parent(numVertices);
        vector<int> rank(numVertices, 0);
        for (int i = 0; i < numVertices; ++i) {
            parent[i] = i;
        }

        vector<pair<int, pair<int, int>>> mstEdges;
        int mstCost = 0;

        for (auto edge : edges) {
            int weight = edge.first;
            int u = edge.second.first;
            int v = edge.second.second;

            if (findParent(u, parent) != findParent(v, parent)) {
                mstEdges.push_back({weight, {u, v}});
                mstCost += weight;
                unionSets(u, v, parent, rank);
            }
        }

        // Print the MST edges
        cout << "Edge \tWeight\n";
        for (auto edge : mstEdges) {
            cout << edge.second.first << " - " << edge.second.second
<< "\t" << edge.first << endl;
        }

        cout << "And total cost of MST is : " << mstCost << endl;
    }

};

int main() {
    int numVertices = 5; // Change this to the number of vertices in
your graph
    Graph graph(numVertices);

    // Adding edges to the graph
    graph.addEdge(0, 1, 2);
    graph.addEdge(0, 3, 6);
    graph.addEdge(1, 2, 3);
    graph.addEdge(1, 3, 8);
    graph.addEdge(1, 4, 5);
    graph.addEdge(2, 4, 7);
    graph.addEdge(3, 4, 9);
```

```
    // Find and print the Minimum Spanning Tree (MST) using Kruskal's
algorithm
    cout << "Minimum Spanning Tree (MST) using Kruskal's
Algorithm:\n";
    graph.kruskalMST();

    return 0;
}
```

Output-