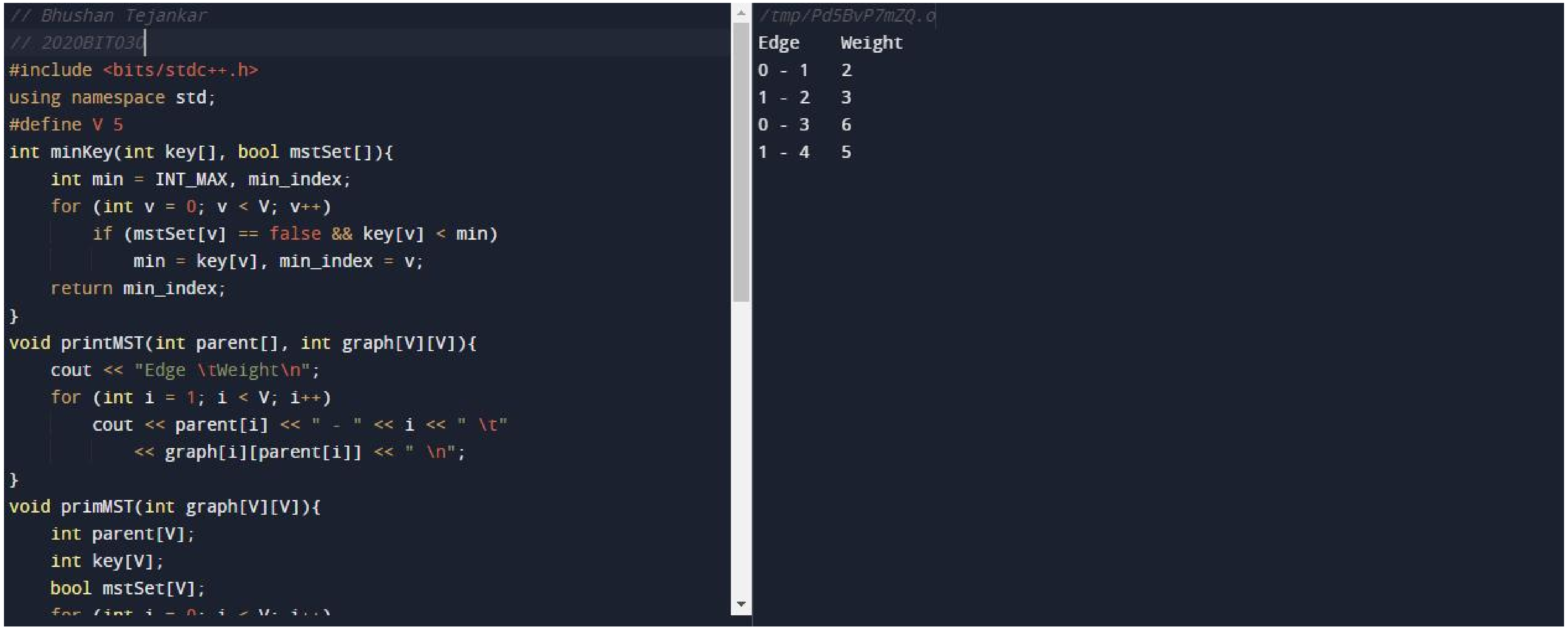
Name : Bhushan Sharad Tejankar

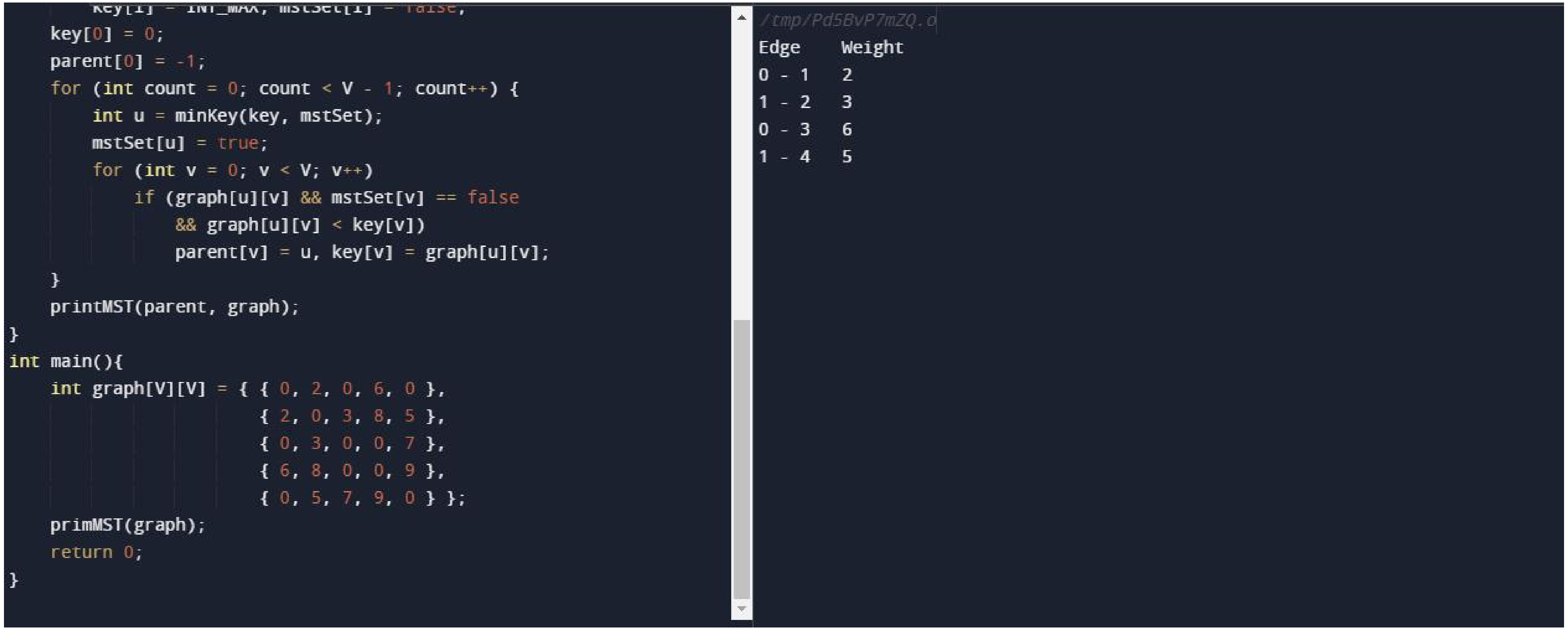
Roll no. : I30

reg_no. : 2020BIT030



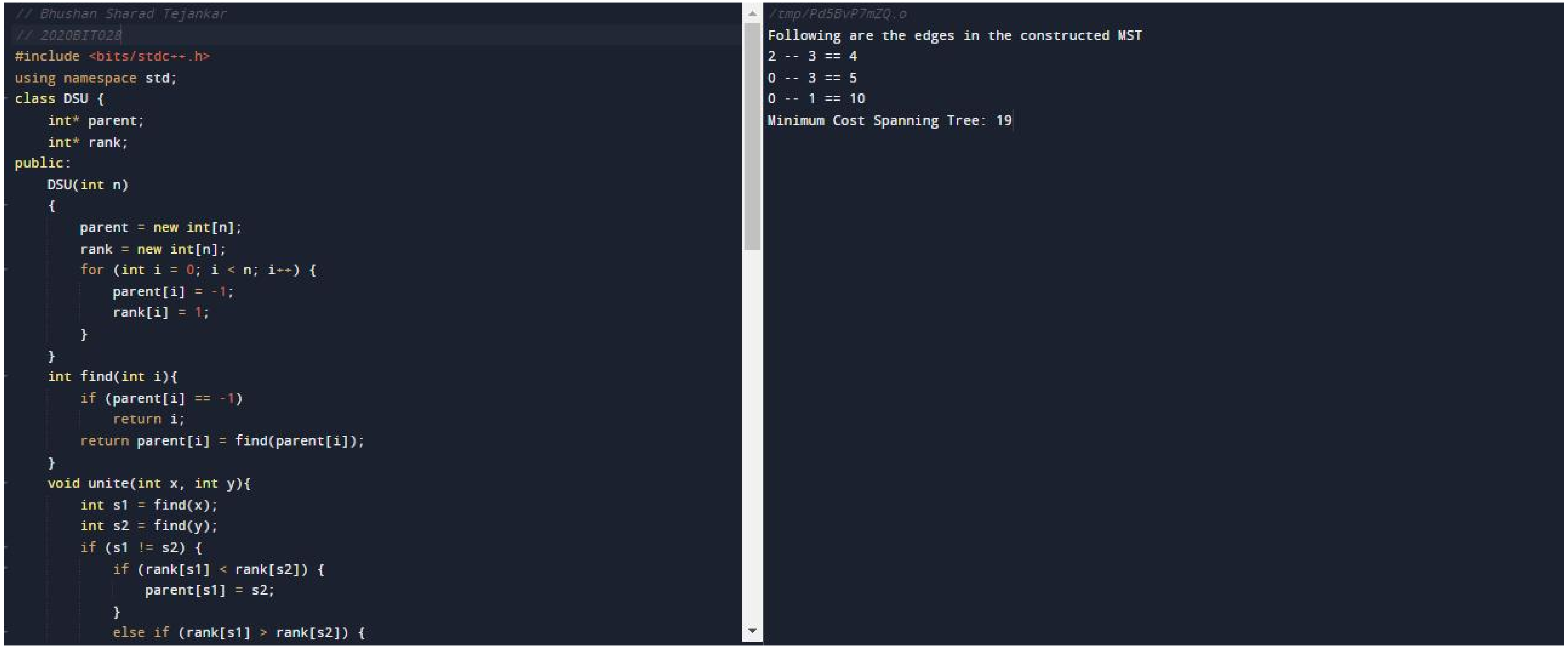
```cpp
// Bhushan Tejankar
// 2020BIT030
#include <bits/stdc++.h>
using namespace std;
#define V 5
int minKey(int key[], bool mstSet[]){
    int min = INT_MAX, min_index;
    for (int v = 0; v < V; v++)
        if (mstSet[v] == false && key[v] < min)
            min = key[v], min_index = v;
    return min_index;
}
void printMST(int parent[], int graph[V][V]){
    cout << "Edge \tWeight\n";
    for (int i = 1; i < V; i++)
        cout << parent[i] << " - " << i << " \t"
            << graph[i][parent[i]] << " \n";
}
void primMST(int graph[V][V]){
    int parent[V];
    int key[V];
    bool mstSet[V];
```

```
/tmp/Pd5BvP7mZQ.o
Edge    Weight
0 - 1   2
1 - 2   3
0 - 3   6
1 - 4   5
```



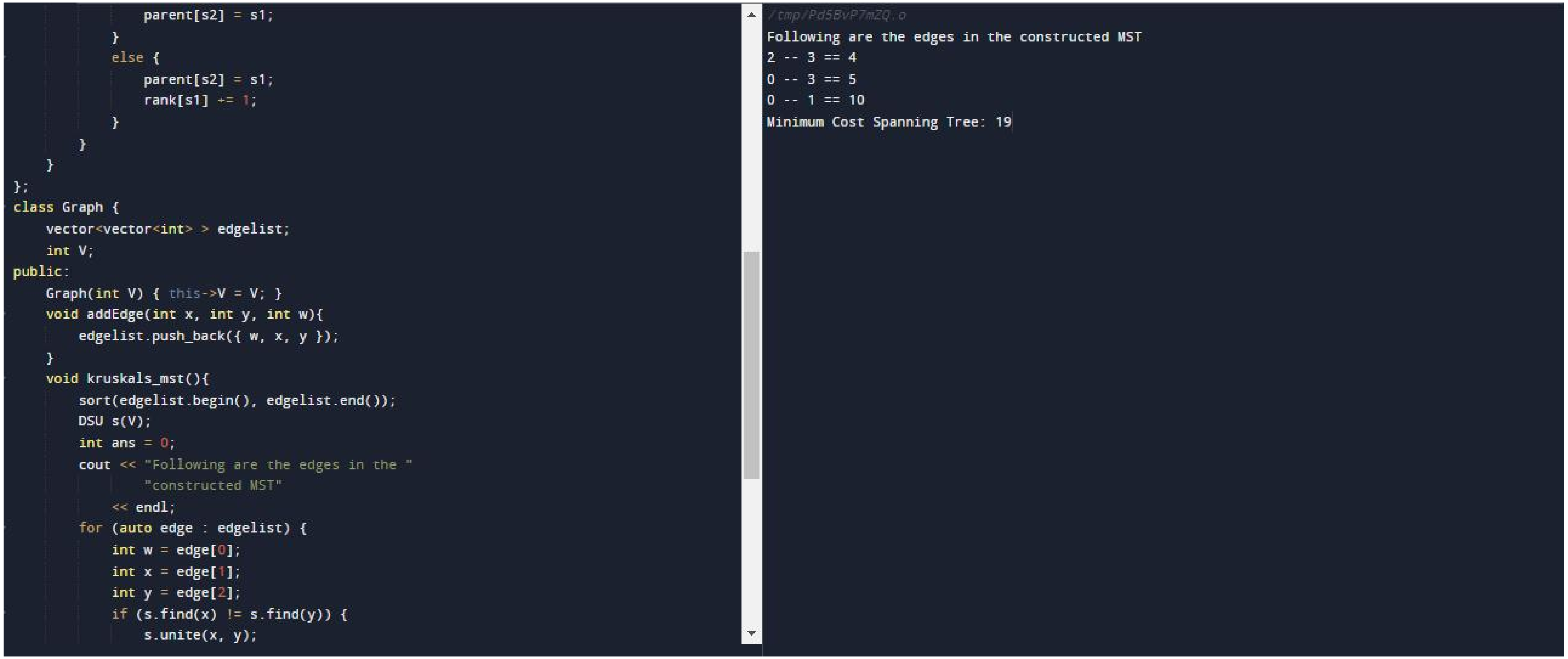
```
        key[i] = INT_MAX, mstSet[i] = false;
    key[0] = 0;
    parent[0] = -1;
    for (int count = 0; count < V - 1; count++) {
        int u = minKey(key, mstSet);
        mstSet[u] = true;
        for (int v = 0; v < V; v++)
            if (graph[u][v] && mstSet[v] == false
                && graph[u][v] < key[v])
                parent[v] = u, key[v] = graph[u][v];
    }
    printMST(parent, graph);
}
int main(){
    int graph[V][V] = { { 0, 2, 0, 6, 0 },
                        { 2, 0, 3, 8, 5 },
                        { 0, 3, 0, 0, 7 },
                        { 6, 8, 0, 0, 9 },
                        { 0, 5, 7, 9, 0 } };
    primMST(graph);
    return 0;
}
```

```
/tmp/Pd5BvP7mZQ.o
Edge    Weight
0 - 1   2
1 - 2   3
0 - 3   6
1 - 4   5
```



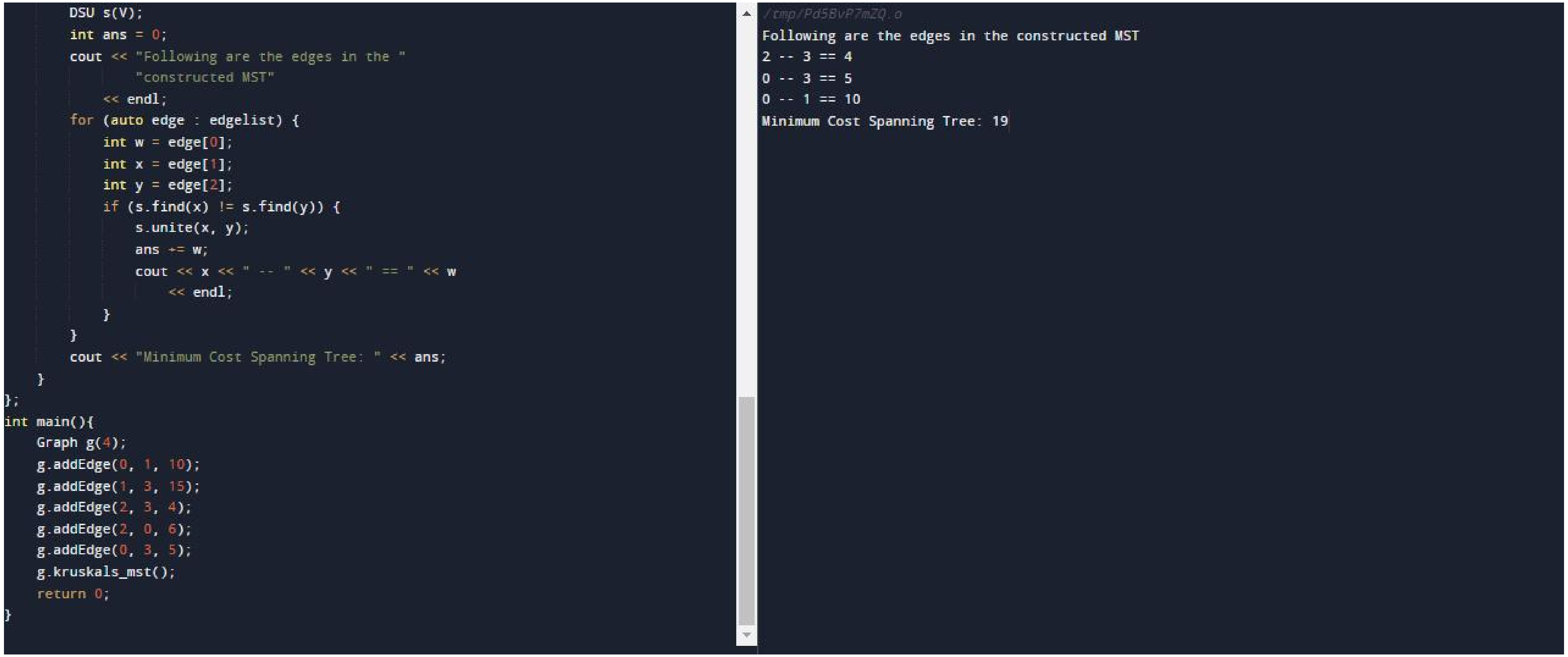
```
// Bhushan Sharad Tejankar
// 2020BIT028
#include <bits/stdc++.h>
using namespace std;
class DSU {
    int* parent;
    int* rank;
public:
    DSU(int n)
    {
        parent = new int[n];
        rank = new int[n];
        for (int i = 0; i < n; i++) {
            parent[i] = -1;
            rank[i] = 1;
        }
    }
    int find(int i){
        if (parent[i] == -1)
            return i;
        return parent[i] = find(parent[i]);
    }
    void unite(int x, int y){
        int s1 = find(x);
        int s2 = find(y);
        if (s1 != s2) {
            if (rank[s1] < rank[s2]) {
                parent[s1] = s2;
            }
            else if (rank[s1] > rank[s2]) {
```

```
/tmp/Pd5BvP7mZQ.o
Following are the edges in the constructed MST
2 -- 3 == 4
0 -- 3 == 5
0 -- 1 == 10
Minimum Cost Spanning Tree: 19
```



```
                parent[s2] = s1;
            }
            else {
                parent[s2] = s1;
                rank[s1] += 1;
            }
        }
    }
};
class Graph {
    vector<vector<int> > edgelist;
    int V;
public:
    Graph(int V) { this->V = V; }
    void addEdge(int x, int y, int w){
        edgelist.push_back({ w, x, y });
    }
    void kruskals_mst(){
        sort(edgelist.begin(), edgelist.end());
        DSU s(V);
        int ans = 0;
        cout << "Following are the edges in the "
                "constructed MST"
            << endl;
        for (auto edge : edgelist) {
            int w = edge[0];
            int x = edge[1];
            int y = edge[2];
            if (s.find(x) != s.find(y)) {
                s.unite(x, y);
```

```
/tmp/Pd5BvP7mZQ.o
Following are the edges in the constructed MST
2 -- 3 == 4
0 -- 3 == 5
0 -- 1 == 10
Minimum Cost Spanning Tree: 19
```



```
        DSU s(V);
        int ans = 0;
        cout << "Following are the edges in the "
                "constructed MST"
            << endl;
        for (auto edge : edgelist) {
            int w = edge[0];
            int x = edge[1];
            int y = edge[2];
            if (s.find(x) != s.find(y)) {
                s.unite(x, y);
                ans += w;
                cout << x << " -- " << y << " == " << w
                    << endl;
            }
        }
        cout << "Minimum Cost Spanning Tree: " << ans;
    }
};
int main(){
    Graph g(4);
    g.addEdge(0, 1, 10);
    g.addEdge(1, 3, 15);
    g.addEdge(2, 3, 4);
    g.addEdge(2, 0, 6);
    g.addEdge(0, 3, 5);
    g.kruskals_mst();
    return 0;
}
```

```
/tmp/Pd5BvP7mZQ.o
Following are the edges in the constructed MST
2 -- 3 == 4
0 -- 3 == 5
0 -- 1 == 10
Minimum Cost Spanning Tree: 19
```

THANK
YOU!