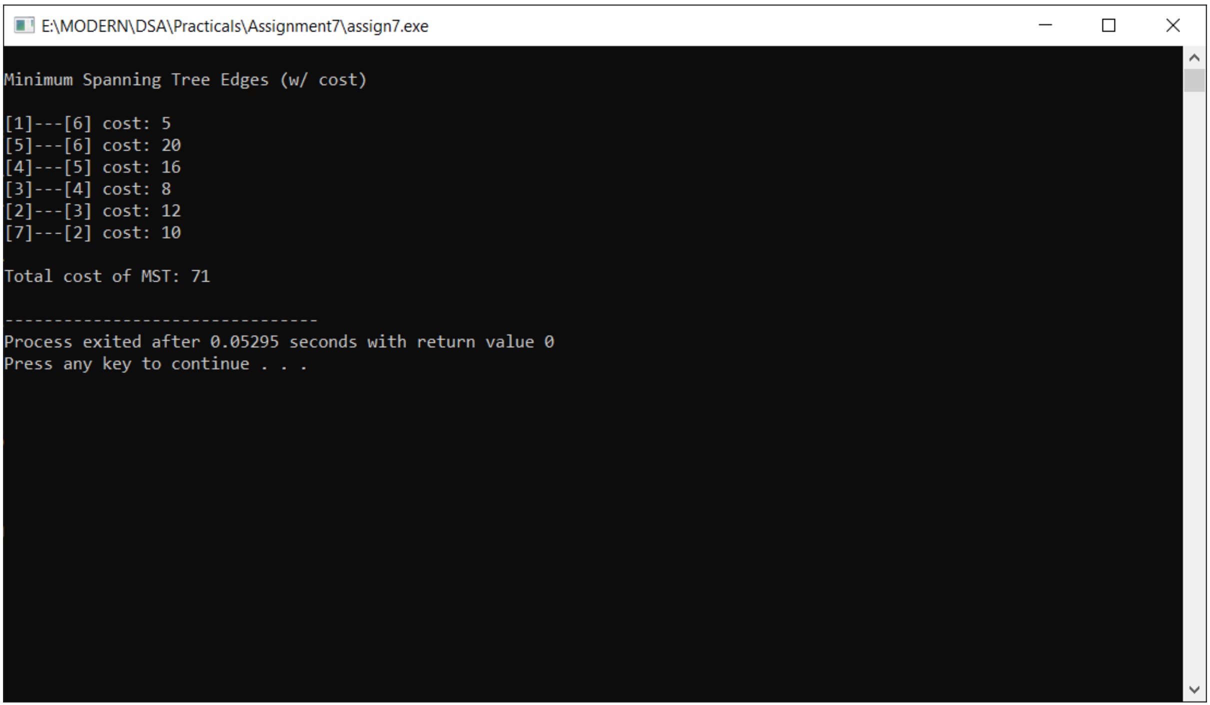
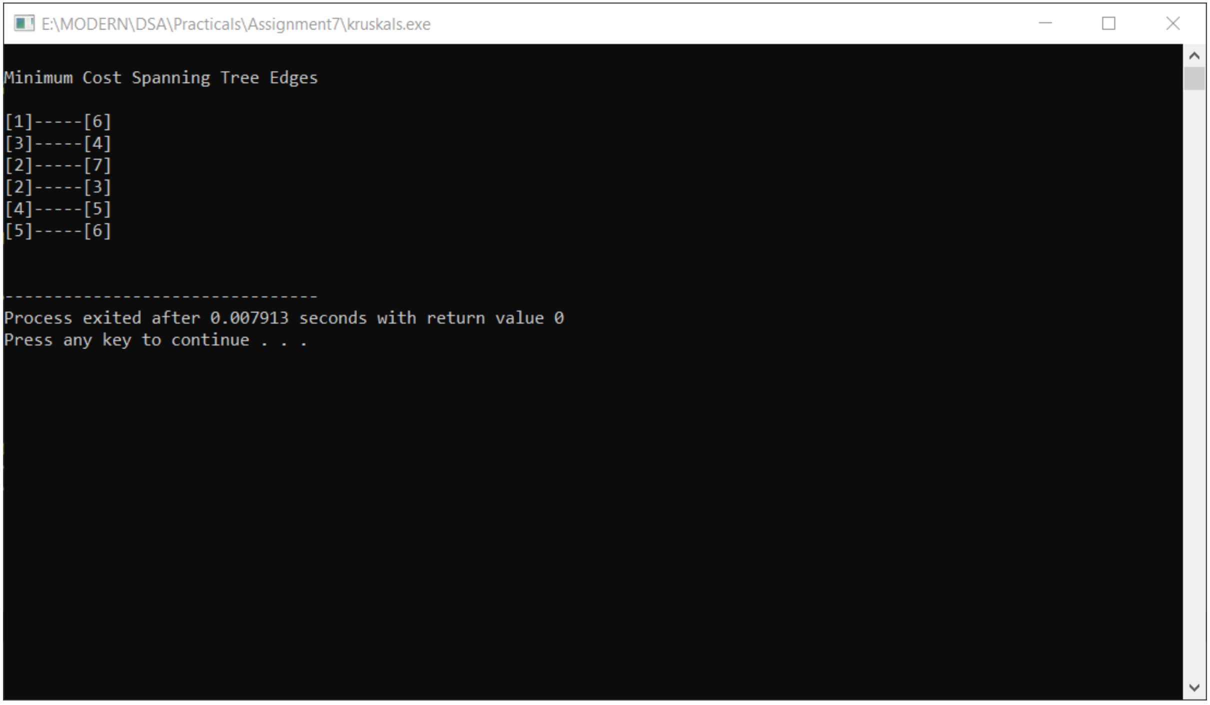
```
 /*
                           ASSIGNMENT NO.7
NAME- ABRAR SHAIKH                               ROLL NO. - 23570
           TOPIC- MST using Prim’s and Kuskal’s Algorithm
*/


#include <iostream>
#define V 8
#define I 32767


using namespace std;


 //********************************* PRIMS ALGORITHM
******************************************
void PrintMST(int T[][V-2], int G[V][V]){
    cout << "\nMinimum Spanning Tree Edges (w/ cost)\n" << endl;
    int sum {0};
    for (int i {0}; i<V-2; i++){
        int c = G[T[0][i]][T[1][i]];
        cout << "[" << T[0][i] << "]---[" << T[1][i] << "] cost: "
<< c << endl;
        sum += c;
    }
    cout << endl;
    cout << "Total cost of MST: " << sum << endl;
}


void PrimsMST(int G[V][V], int n){
    int u;
    int v;
    int min {I};
    int track [V];
    int T[2][V-2] {0};

    // Initial: Find min cost edge
    for (int i {1}; i<V; i++){
        track[i] = I;  // Initialize track array with INFINITY
        for (int j {i}; j<V; j++){
            if (G[i][j] < min){
                min = G[i][j];
                u = i;
                v = j;
            }
        }
    }
    T[0][0] = u;
    T[1][0] = v;
    track[u] = track[v] = 0;

    // Initialize track array to track min cost edges
    for (int i {1}; i<V; i++){
        if (track[i] != 0){
            if (G[i][u] < G[i][v]){
                track[i] = u;
            } else {
                track[i] = v;
            }
        }
    }

    // Repeat
    for (int i {1}; i<n-1; i++){
        int k;
        min = I;
        for (int j {1}; j<V; j++){
            if (track[j] != 0 && G[j][track[j]] < min){
                k = j;
                min = G[j][track[j]];
            }
        }
        T[0][i] = k;
        T[1][i] = track[k];
        track[k] = 0;

        // Update track array to track min cost edges
        for (int j {1}; j<V; j++){
            if (track[j] != 0 && G[j][k] < G[j][track[j]]){
                track[j] = k;
            }
        }
    }
    PrintMST(T, G);
}

int main()
{
     int cost [V][V]
     {
            {I, I, I, I, I, I, I, I},
            {I, I, 25, I, I, I, 5, I},
            {I, 25, I, 12, I, I, I, 10},
            {I, I, 12, I, 8, I, I, I},
            {I, I, I, 8, I, 16, I, 14},
            {I, I, I, I, 16, I, 20, 18},
            {I, 5, I, I, I, 20, I, I},
            {I, I, 10, I, 14, 18, I, I},
    };

    int n = sizeof(cost[0])/sizeof(cost[0][0]) - 1;

    PrimsMST(cost, n);
    return 0;
}

#include <iostream>

#define I 32767  // Infinity
#define V 7  // # of vertices in Graph
#define E 9  // # of edges in Graph

using namespace std;

void PrintMCST(int T[][V-1], int A[][E]){
    cout << "\nMinimum Cost Spanning Tree Edges\n" << endl;
    for (int i {0}; i<V-1; i++){
        cout << "[" << T[0][i] << "]-----[" << T[1][i] << "]" <<
endl;
    }
    cout << endl;
}

// Set operations: Union and Find
```

```
void Union(int u, int v, int s[]){
    if (s[u] < s[v]){
        s[u] += s[v];
        s[v] = u;
    } else {
        s[v] += s[u];
        s[u] = v;
    }
}

int Find(int u, int s[]){
    int x = u;
    int v = 0;

    while (s[x] > 0){
        x = s[x];
    }

    while (u != x){
        v = s[u];
        s[u] = x;
        u = v;
    }
    return x;
}

void KruskalsMCST(int A[3][9]){
    int T[2][V-1];  // Solution array
    int track[E] {0};  // Track edges that are included in solution
    int set[V+1] = {-1, -1, -1, -1, -1, -1, -1, -1};  // Array for
finding cycle

    int i {0};
    while (i < V-1){
        int min = I;
        int u {0};
        int v {0};
        int k {0};

        // Find a minimum cost edge
        for (int j {0}; j<E; j++){
            if (track[j] == 0 && A[2][j] < min){
                min = A[2][j];
                u = A[0][j];
                v = A[1][j];
                k = j;
            }
        }

        // Check if the selected min cost edge (u, v) forming a
cycle or not
        if (Find(u, set) != Find(v, set)){
            T[0][i] = u;
            T[1][i] = v;

            // Perform union
            Union(Find(u, set), Find(v, set), set);
            i++;
        }
        track[k] = 1;
    }

    PrintMCST(T, A);
}

int main() {
    int edges[3][9] = {{ 1, 1,  2,  2, 3,  4,  4,  5,  5},
                       { 2, 6,  3,  7, 4,  5,  7,  6,  7},
                       {25, 5, 12, 10, 8, 16, 14, 20, 18}};

    KruskalsMCST(edges);

    return 0;
}
```



```
Minimum Spanning Tree Edges (w/ cost)

[1]---[6] cost: 5
[5]---[6] cost: 20
[4]---[5] cost: 16
[3]---[4] cost: 8
[2]---[3] cost: 12
[7]---[2] cost: 10

Total cost of MST: 71

--------------------------------
Process exited after 0.05295 seconds with return value 0
Press any key to continue . . .
```



```
Minimum Cost Spanning Tree Edges

[1]-----[6]
[3]-----[4]
[2]-----[7]
[2]-----[3]
[4]-----[5]
[5]-----[6]


--------------------------------
Process exited after 0.007913 seconds with return value 0
Press any key to continue . . .
```

GitHub Repository- https://github.com/abssha/DSA.git