Assignment 6

```
#include <iostream>
#include <climits>
using namespace std;

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

int findMinKey(int key[], bool mstSet[], int V) {
    int min = INT_MAX, minIndex;

    for (int v = 0; v < V; v++)
        if (!mstSet[v] && key[v] < min)
            min = key[v], minIndex = v;

    return minIndex;
}

void printMST(int parent[], Edge edges[], int V) {
    int totalCost = 0;
    cout << "\nEdges in the Minimum Spanning Tree:\n";
    for (int i = 1; i < V; i++) {
        for (int j = 0; j < V - 1; j++) {
            if ((edges[j].u == parent[i] && edges[j].v == i) || (edges[j].u == i && edges[j].v == parent[i])) {
                cout << "[" << parent[i] + 1 << "] ----- [" << i + 1 << "] : " << edges[j].weight << endl;
                totalCost += edges[j].weight;
                break;
            }
        }
    }
```

```
    cout << "\nTotal weight of the Minimum Spanning Tree: " << totalCost << endl;
}

void primMST(Edge edges[], int V, int E) {
    int parent[V];     // Array to store the MST
    int key[V];        // To store minimum weight edge for each vertex
    bool mstSet[V];    // To track vertices included in the MST

    // Initialize key values as infinite and mstSet[] as false
    for (int i = 0; i < V; i++) {
        key[i] = INT_MAX;
        mstSet[i] = false;
    }

    key[0] = 0;        // Start from the first vertex
    parent[0] = -1;    // First node is the root of the MST

    // Construct the MST
    for (int count = 0; count < V - 1; count++) {
        // Pick the minimum key vertex from the set of vertices not yet included in the MST
        int u = findMinKey(key, mstSet, V);

        mstSet[u] = true;  // Include the picked vertex in the MST

        // Update the key and parent values for the adjacent vertices
        for (int i = 0; i < E; i++) {
            if ((edges[i].u == u || edges[i].v == u)) {
                int v = (edges[i].u == u) ? edges[i].v : edges[i].u;
                if (!mstSet[v] && edges[i].weight < key[v]) {
                    parent[v] = u;
                    key[v] = edges[i].weight;
```

```
            }
          }
        }
    }

    // Print the constructed MST
    printMST(parent, edges, V);
}

int main() {
    int V, E;

    // Input number of vertices and edges
    cout << "Enter the number of vertices: ";
    cin >> V;
    cout << "Enter the number of edges: ";
    cin >> E;

    Edge edges[E];  // Array to store all edges

    // Input edges in the format (u, v, weight)
    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;
        edges[i].u--;  // Convert to zero-based indexing
        edges[i].v--;  // Convert to zero-based indexing
    }

    // Run Prim's algorithm
    primMST(edges, V, E);
```

```
    return 0;
}
```

Output: