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

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

class Graph {
    int V;
    vector<vector<int>> adjMatrix;
    vector<Edge> edges;

public:
    Graph(int vertices) : V(vertices) {
        adjMatrix = vector<vector<int>>(V, vector<int>(V, 0));
    }

    void addEdge(int u, int v, int weight) {
        adjMatrix[u][v] = weight;
        adjMatrix[v][u] = weight;
        edges.push_back({u, v, weight});
    }

    void kruskalMST() {
        vector<int> parent(V, -1);
        auto find = [&](int i) {
            while (parent[i] != -1) i = parent[i];
            return i;
        };

        auto unionSet = [&](int u, int v) {
            parent[find(u)] = find(v);
        };

        sort(edges.begin(), edges.end(), [](Edge a, Edge b) {
            return a.weight < b.weight;
        });

        int totalWeight = 0;
        cout << "Kruskal's MST:" << endl;
        for (Edge &e : edges) {
            if (find(e.u) != find(e.v)) {
                cout << e.u << " - " << e.v << " : " << e.weight << endl;
                totalWeight += e.weight;
                unionSet(e.u, e.v);
            }
        }
        cout << "Total Weight: " << totalWeight << endl;
    }

    void primMST() {
```

```
        vector<int> key(V, INT_MAX);
        vector<bool> inMST(V, false);
        vector<int> parent(V, -1);

        key[0] = 0;

        for (int i = 0; i < V - 1; i++) {
            int u = -1;
            for (int v = 0; v < V; v++) {
                if (!inMST[v] && (u == -1 || key[v] < key[u])) {
                    u = v;
                }
            }

            inMST[u] = true;

            for (int v = 0; v < V; v++) {
                if (adjMatrix[u][v] && !inMST[v] && adjMatrix[u][v] <
key[v]) {
                    key[v] = adjMatrix[u][v];
                    parent[v] = u;
                }
            }
        }

        int totalWeight = 0;
        cout << "Prim's MST:" << endl;
        for (int v = 1; v < V; v++) {
            cout << parent[v] << " - " << v << " : " <<
adjMatrix[parent[v]][v] << endl;
            totalWeight += adjMatrix[parent[v]][v];
        }
        cout << "Total Weight: " << totalWeight << endl;
    }
};

int main() {
    int V = 6; // Number of departments/nodes
    Graph g(V);

    g.addEdge(0, 1, 4);
    g.addEdge(0, 2, 6);
    g.addEdge(1, 2, 5);
    g.addEdge(1, 3, 3);
    g.addEdge(2, 3, 8);
    g.addEdge(3, 4, 2);
    g.addEdge(4, 5, 7);
    g.addEdge(3, 5, 9);

    g.kruskalMST();
    g.primMST();

    return 0;
}
```