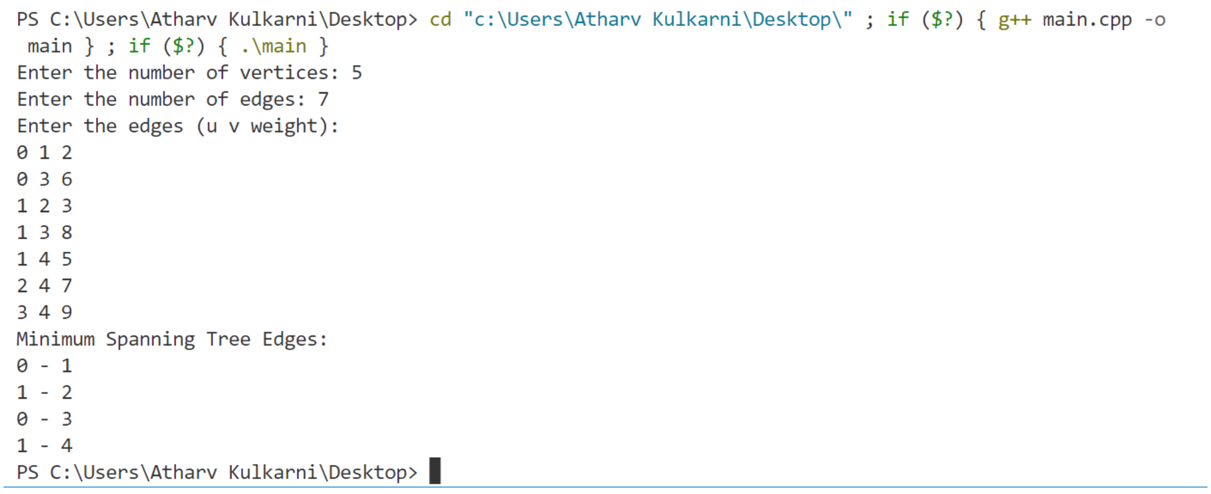
```cpp
#include <iostream>
#include <vector>
#include <queue>

using namespace std;

#define INF 0x3f3f3f3f

typedef pair<int, int> pii;

class Graph {
    int V;
    vector<vector<pii>> adj;

public:
    Graph(int V)
    {
        this->V = V;
        adj.resize(V);
    }

    void addEdge(int u, int v, int weight)
    {
        adj[u].push_back(make_pair(v, weight));
        adj[v].push_back(make_pair(u, weight));
    }

    void dijkstraMST()
    {
        vector<int> key(V, INF); // Key values used to pick minimum weight
edge in cut
        vector<int> parent(V, -1); // Array to store constructed MST
        vector<bool> inMST(V, false); // To represent set of vertices included
in MST

        priority_queue<pii, vector<pii>, greater<pii>> pq; // Min Heap

        // Start with vertex 0
        int src = 0;
        pq.push(make_pair(0, src));
        key[src] = 0;

        while (!pq.empty()) {
            int u = pq.top().second;
            pq.pop();

            inMST[u] = true;
            // Traverse all adjacent vertices of u
            for (auto i = adj[u].begin(); i != adj[u].end(); ++i) {
                int v = (*i).first;
                int weight = (*i).second;

                // If v is not yet included in MST and weight of (u,v) is
smaller than current key of v
                if (inMST[v] == false && key[v] > weight) {
                    // Updating key of v
                    key[v] = weight;
                    pq.push(make_pair(key[v], v));
                    parent[v] = u;
                }
            }
        }

        // Print edges of MST
        cout << "Minimum Spanning Tree Edges:" << endl;
        for (int i = 1; i < V; ++i)
            cout << parent[i] << " - " << i << endl;
    }
};

int main()
{
    int V, E;
    cout << "Enter the number of vertices: ";
    cin >> V;
    cout << "Enter the number of edges: ";
    cin >> E;

    Graph g(V);

    cout << "Enter the edges (u v weight):" << endl;
    for (int i = 0; i < E; ++i) {
        int u, v, weight;
        cin >> u >> v >> weight;
        g.addEdge(u, v, weight);
    }

    g.dijkstraMST();

    return 0;
}
```



```
PS C:\Users\Atharv Kulkarni\Desktop> cd "c:\Users\Atharv Kulkarni\Desktop\" ; if ($?) { g++ main.cpp -o
 main } ; if ($?) { .\main }
Enter the number of vertices: 5
Enter the number of edges: 7
Enter the edges (u v weight):
0 1 2
0 3 6
1 2 3
1 3 8
1 4 5
2 4 7
3 4 9
Minimum Spanning Tree Edges:
0 - 1
1 - 2
0 - 3
1 - 4
PS C:\Users\Atharv Kulkarni\Desktop>
```