RAJSHAHI UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Lab report: 07
Course No.: CSE 2202

Date of Experiment: 14.01.2019
Date of Submission: 21.01.2019

Submitted to:
Biprodip Pal
Assistant Professor,
Department of Computer Science and Engineering
Rajshahi University of Engineering and Technology

Submitted by:
Riyad Morshed Shoeb
Roll No: 1603013
Section: A
Department of Computer Science and Engineering
Rajshahi University of Engineering and Technology

Problem: Implement Kruskal's Algorithm for finding Minimum Spanning Tree for a weighted undirected graph

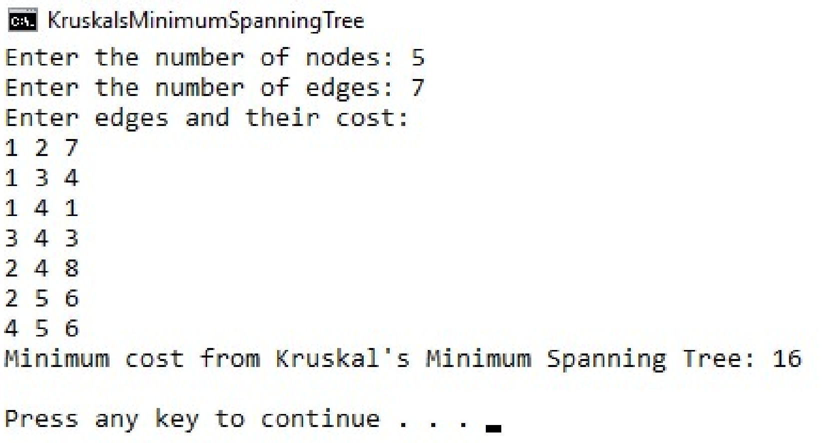
Approach:
-Use disjoint set to implement union and set finding operations. These operations should use the concept of parent child relationship to represent element of a set.

Code:

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <vector>
#include <set>
#include <iterator>
#include <algorithm>
using namespace std;


//class definition
class EDGE
{
public:
	int vertex1;
	int vertex2;
	int cost;
	EDGE(int, int, int);
	//operator overloading for this class
	bool  operator<  (const  EDGE  &edge)const  {return  cost  <
edge.cost;}
};

EDGE::EDGE(int u, int v, int c)
{
	vertex1 = u;
	vertex2 = v;
	cost = c;
}


//global variables
vector <EDGE> edges;
int parent[100];
int NumberOfNodes;
int NumberOfEdges;


//function definition
void AddEdge(int vertex1, int vertex2, int cost)
{
	edges.push_back(EDGE(vertex1, vertex2, cost));
}

int FindParent(int node)
```

```
{
      if(parent[node] == node)
            return node;
      return parent[node] = FindParent(parent[node]);
}

int KruskalsSpanningTree(void)
{
      int iter;
      int SpanningTreeCost = 0;
      int ParentOfVertex1;
      int ParentOfVertex2;

      sort(edges.begin(), edges.end());

      for(iter=0; iter<NumberOfNodes; iter++)
            parent[iter] = iter;

      for(iter=0; iter<NumberOfEdges; iter++)
      {
          ParentOfVertex1 = FindParent(edges[iter].vertex1);
          ParentOfVertex2 = FindParent(edges[iter].vertex2);
          if(ParentOfVertex1 != ParentOfVertex2)
            {
              parent[ParentOfVertex1] = ParentOfVertex2;
              SpanningTreeCost += edges[iter].cost;
        }
    }

      return SpanningTreeCost;
}


//main function
int main(void)
{
      int iter;
      int vertex1;
      int vertex2;
      int cost;

      cout << "Enter the number of nodes: ";
      cin >> NumberOfNodes;
      cout << "Enter the number of edges: ";
      cin >> NumberOfEdges;
      cout << "Enter edges and their cost: " << endl;
      for(iter=0; iter<NumberOfEdges; iter++)
      {
            cin >> vertex1 >> vertex2 >> cost;
            AddEdge(vertex1, vertex2, cost);
      }

      cout << "Minimum cost from Kruskal's Minimum Spanning Tree: "
<< KruskalsSpanningTree() << endl;
}
```

Output:



KruskalsMinimumSpanningTree

```
Enter the number of nodes: 5
Enter the number of edges: 7
Enter edges and their cost:
1 2 7
1 3 4
1 4 1
3 4 3
2 4 8
2 5 6
4 5 6
Minimum cost from Kruskal's Minimum Spanning Tree: 16

Press any key to continue . . .
```