IMPORTANT NOTE (PLEASE READ):

For the purposes of simplifying and streamlining the development and implementation of of the functionalities in the relevant USs, our team (G123) decided to create a class that would be used throughout the project to represent the edges of any graph. This class was given the name “Connection”, and the following code block displays its source code.

---CODE BLOCK START---

```
final class Connection {

        public int origin;

        public int end;

        public int value;

        public Connection(int origin, int end, int value){

                this.origin = origin;

                this.end = end;

                this.value = value;

        }

}
```

---CODE BLOCK END---

We believe that that usage of this class does not violate any of the rules established for the implementation of a solution to US013. This class is merely a structure created to simplify the storage of graphs and does not perform any function other than such. We believe therefore believe that this does not qualify as a non-primitive operation, and does not break any rule outlined in the assignment. Regardless, on to the implementations of algorithms:

The following is the source code for the implementation of a method that groups together the sorting and Kruskal algorithms:

```
-----------------------------------------CODE BLOCK START-----------------------------------------
public static ArrayList<Connection> US13List(ArrayList<Connection> connectionList,
ArrayList<String> nodeList){
    sortConnectionList(connectionList);
    connectionList = kruskalAlgorithmPrelistedNodes(connectionList, nodeList);
    return connectionList;
}
-----------------------------------------CODE BLOCK END-----------------------------------------
```

The following is the source code for the implementation of the sorting algorithm (BubbleSort) used in this project:

```
-----------------------------------------CODE BLOCK START-----------------------------------------
public static ArrayList<Connection> sortConnectionList(ArrayList<Connection>
connectionList){
  for(int i = connectionList.size()-1; i >= 0; i--){
        for(int j = 1; j <= i; j++){
                if(connectionList.get(j-1).value > connectionList.get(j).value){
                        Connection tempConn = connectionList.get(j-1);
                        connectionList.set(j-1, connectionList.get(j));
                        connectionList.set(j, tempConn);
                }
        }
  }
   return connectionList;
}
-----------------------------------------CODE BLOCK END-----------------------------------------
```

The following is the source code for the implementation of the Kruskal Algorithm used in this project:

---CODE BLOCK START---

```
public static ArrayList<Connection> kruskalAlgorithmPrelistedNodes
(ArrayList<Connection> list, ArrayList<String> nodes){

    ArrayList<Integer> nodeSets = new ArrayList<Integer>();

    int listEdgeCounter = 0;

    for(int i = 0; i < nodes.size(); i++) { nodeSets.add(-1); }

    ArrayList<Connection> map = new ArrayList<Connection>();

    while(map.size() < nodes.size()-1){

        int originParentIndex = list.get(listEdgeCounter).origin, endParentIndex =
        list.get(listEdgeCounter).end;

        int originSet = nodeSets.get(originParentIndex), endSet =
        nodeSets.get(endParentIndex);

        while((originSet != -1 || endSet != -1) && originSet != endSet){

                    if(originSet != -1) { originParentIndex = originSet; originSet =
                    nodeSets.get(originParentIndex); }

                    if(endSet != -1) { endParentIndex = endSet; endSet =
                    nodeSets.get(endParentIndex); }

        }

        if(endSet == -1 && originParentIndex != endParentIndex){

                nodeSets.set(endParentIndex, originParentIndex);

                map.add(list.get(listEdgeCounter));

        }else if(originSet == -1 && originParentIndex != endParentIndex){

                nodeSets.set(originParentIndex, endParentIndex);

                map.add(list.get(listEdgeCounter));

        }

        listEdgeCounter++;

    }

    return map;

}
```

---CODE BLOCK END---