Problem 1

Edge

```
package hw4;

import java.util.Comparator;
import java.util.Objects;

/** <b>Edge</b> represents an <b>immutable</b> edge in the graph
 * it set up a bridge between two node, and can be labeled with a name
 * There many kinds of edge, include reflective edge, which the Node "from"
and "to"
 *  reference to the same node
 * An edge can be compared by its name
 */
public class Edge implements Comparable<Edge> {
    private final String from;
    private final String  to;
    private final String name;

    // Abstraction Function:
    // an edge that start from node "from" and ends at Node "to"
    // named with String "name"

    // Representation invariant for every Edge:
    // from != NULL && to != NULL && name != NULL

    /** @param a is the String represent node that  the edge start from
     * @param b is the String represent node that the edge ends at
     * @param n is the String represent the label of the edge
          @effects Constructs a new Edge that connects Node a and Node b
         with name n
     */
    public Edge (String a, String b, String n){
        throw new RuntimeException("Edge with Label Constructor not
implemented");
    }

    /** @param a is the String represent node that  the edge start from
     * @param b is the String represent node that the edge ends at
     @effects Constructs a new Edge that connects Node a and Node b
     with name n
     */
    public Edge (String a, String b) {
            throw new RuntimeException("Edge Constuctor not implemented");
    }


    /** Returns the start from node
```

```
            @return source node of this edge
     */
    public String getFrom() {
        throw new RuntimeException("getFrom  not implemented");
    }

    /**
     *  Returns the end node of the edge
     * @return Node Destination of this edge
     */
    public String getTo() {
        throw new RuntimeException("getTo  not implemented");
    }

    /**
     * Returns the name of the edge
     * @return String the name of this edge
     */
    public String getName() {
       throw new RuntimeException("getName not implemented");
    }

    /** Compares two Edges
     @param edge The Edge to be compared.
     @requires rn != null
     @return positive if  this.name > edge.name
     0 if the name are same
     negative if this.name < edge.name
     */
    @Override
    public int compareTo(Edge edge) {
           throw new RuntimeException("compareTo not implemented");
    }

    /**
    @param o The Edge to be compared.
     @return true iff this has the same attribute as o
     */
    @Override
    public boolean equals(Object o) {
        throw new RuntimeException("Equals not implemented");
    }

    /**
     @return int the hashcode of edge
     */
    @Override
    public int hashCode() {
       throw new RuntimeException("HashCode not implemented");
    }
}
```

Graph

```
package hw4;

import java.util.*;

/**
 * <b>Graph</b> represents an <b>mutable</b> graph consists of nodes and
edges
 * It represents a multi-bidirectional graph
 */
public class Graph {
    private TreeMap<String, TreeSet<Edge>> graph;

    // Abstraction Function:
    // Make use of adjacent list, we have a map that represent the start
nodes corresponded with
    // a set of edges connected with it

    // Representation invariant for every Graph g:
    // forall list of graph.values() :: forall edge in list ::
graph.keys().contains(edge.getTo())
    // An empty graph is allowed

    /**
     * @effects Construct an empty graph
     */
    public Graph() {
        throw new RuntimeException("Graph Constructor not Implemented");
    }

    /**
     * add new node to the graph
     * @param a String represent Node to be added
     * @return boolean true iff the edge successfully added to the graph
     */
    public boolean addNode(String a) {
        throw new RuntimeException("addNode Constructor not Implemented");
    }

    /**
     * get all nodes in the graph
     * @return Set of Strings that represent nodes
     * An empty set will return if the graph is empty
     */
    public Set<String> getNodes() {
       throw new RuntimeException("getNode Constructor not Implemented");
    }

    /**
     * add new nodes and a edge to the graph
     * @param a String represent Node which the edge starts at
```

```
     * @param b String represent Node which the edge ends at
     * @param edgeName String the name of the edge
     * @return boolean true iff the edge successfully added to the graph
     */
    public boolean connect(String a, String b, String edgeName) {
        throw new RuntimeException("connect with edge not Implemented");
    }

    /**
     * add new nodes and a edge to the graph
     * @param a String represent Node which the edge starts at
     * @param b String represent Node which the edge ends at
     * @return boolean true iff the edge successfully added to the graph
     */
    public boolean connect(String a, String b) {
       throw new RuntimeException("connect not Implemented");
    }

    /**
     * Return a Set of edge that start from given Node a
     * @param a String represent Node where the edge start from
     * @return A set of edge that start from given Node a.
     *  An empty set will return if node "a" does not exist
     */
    public Set<Edge> connectedEdge(String a) {
        throw new RuntimeException("connectEdge Constructor not
Implemented");
    }

    /**
     * Return a Set of edge that start from given Node a
     * @param a String represent Node where the edge start from
     * @return A list of node that start from given Node a, alphabetically
ordered.
     *  An empty list will return if node "a" does not exist
     */
    public List<String> connectedNodes(String a) {
        throw new RuntimeException("connetedNodes not Implemented");
    }
}
```