Faculty of Engineering,

University of Jaffna

Department of Computer Engineering

EC5080: Software Construction

Lab 04

You are given a java program to read the pair of connected city names. Function findAnyRouteToCity(String source, String destination) uses the graph to return a path if possible between the source and destination, or "Not Reachable" in case no such path exists.

E.g.,

When your source is City1 and destination is city 3,

Input:

City1,city2

City3,city4

City3,city5

City2,city3

Press enter

Output:??

Read the line of codes and carefully understand the usage of the graph with the help of the Java collection framework and data structures you learned so far. Paste the output for different inputs.

```
import java.util.HashMap;

import java.util.HashSet;

import java.util.LinkedList;

import java.util.Set;

import java.util.Map;

import java.util.Queue;

import java.util.Scanner;
```

```
class Road {

public String city1;

public String city2;


public Road(String city1, String city2) {

this.city1 = city1;

this.city2 = city2;

}


}


class RoadMap {

Map<String, Set<Road>> roadMap = new HashMap<String, Set<Road>>();


public Set<String> getAllCities() {

return this.roadMap.keySet();

}


public void readLine(String line) {

String[] csv = line.split(",");

String city1 = csv[0];

String city2 = csv[1];

addRoad(city1, city2);
```

```
}

private void addCity(String city) {
this.roadMap.put(city, new HashSet<Road>());
}

private void addRoad(String city1, String city2) {
Road road1 = new Road(city1, city2);
Road road2 = new Road(city2, city1);
if (!this.roadMap.containsKey(city1)) {
addCity(city1);
}
if (!this.roadMap.containsKey(city2)) {
addCity(city2);
}
this.roadMap.get(city1).add(road1);
this.roadMap.get(city2).add(road2);
}

public Set<Road> getAllOutgoingRoads(String node) {
return this.roadMap.get(node);
}

}
```

```
public class GraphAssignment {

static RoadMap roadMap = new RoadMap();


public static void readMap(Scanner scanner) {

while (true) {

String mapLine = scanner.nextLine();

if (mapLine.equals("")) {

break;

}

roadMap.readLine(mapLine);

}

System.out.println("Read map");


}


// Performing BFS to find if destination is reachable from source

public static void findAnyRouteToCity(String source, String destination) {

Map<String, Boolean> visited = new HashMap<String, Boolean>();

Map<String, String> route = new HashMap<String, String>();

Queue<String> queue = new LinkedList<String>();


for (String city : roadMap.getAllCities()) {

visited.put(city, false);
```

```
}
visited.put(source, true);
queue.add(source);
while (!queue.isEmpty()) {
String currentCity = queue.remove();
for (Road road : roadMap.getAllOutgoingRoads(currentCity)) {
if (!visited.get(road.city2)) {
queue.add(road.city2);
visited.put(road.city2, true);
route.put(road.city2, currentCity);
}
}
}
if (!visited.get(destination)) {
System.out.println("Not Reachable");
} else {
String currentCity = destination;
String path = destination;
while (!currentCity.equals(source)) {
currentCity = route.get(currentCity);
path = currentCity + " -> " + path;
}
System.out.println(path);
}
```

```
 }

 public static void main(String[] args) {
 Scanner scanner = new Scanner(System.in);
 readMap(scanner);
 findAnyRouteToCity("Jaffna", "Ariviyalnagar");
 }

}
```

Now write a function to find the shortest path. Here the given input information of connected cities with the report values is stored as a graph. You may use any data structures and/or Java collections.

E.g.,

When your source is City1 and destination is city 3,

Input:

City1,city2,10

City1,city4,20

City4,city3,10

City2,city3,30

Press enter

Output: city1->city4->city3

Paste your output for different inputs.

Upload the zip folder with your Reg_No.