HACETTEPE UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING
BBM204
PROGRAMMING ASSIGNMENT #3

Subject **:** Graphs
Submission Date **:** 22.04.2019
Deadline : 10.05.2019
Programming Language: Java

Problem

In this assignment, you will practice about directed graphs and you will implement a traveler problem. Imagine that you are a traveler and you want to go from a city to another city. In this problem, you have several choices for the transportation type. There may not be a direct path from the source city to the destination city, and thus you have to pass some other cities to reach the destination. Passing through the cities, you need to choose the transportation type of the path connecting these cities.

In your assignment, you assume that the following transportation types are available:

Transportation Types

- Highway
- Airway
- Railway

Program Operations

Part I

You will get the input file named `transportation_network.inp` that includes the adjacency matrixes of cities for three transportation types. The format of this file is as follows:

```
Highway
Cities City1 City2 … CityN
City1 0/1 0/1 … 0/1
City2 0/1 0/1 … 0/1
.
.
CityN 0/1 0/1 … 0/1
Airway
Cities City1 City2 … CityN
City1 0/1 0/1 … 0/1
City2 0/1 0/1 … 0/1
.
.
CityN 0/1 0/1 … 0/1
Railway
Cities City1 City2 … CityN
City1 0/1 0/1 … 0/1
City2 0/1 0/1 … 0/1
.
.
CityN 0/1 0/1 … 0/1
```

Consider the transportation network illustrated in Figure 1. There are 8 cities which are connected to each other with one or more transportation types. It is guaranteed that the graph is directed acyclic graph.

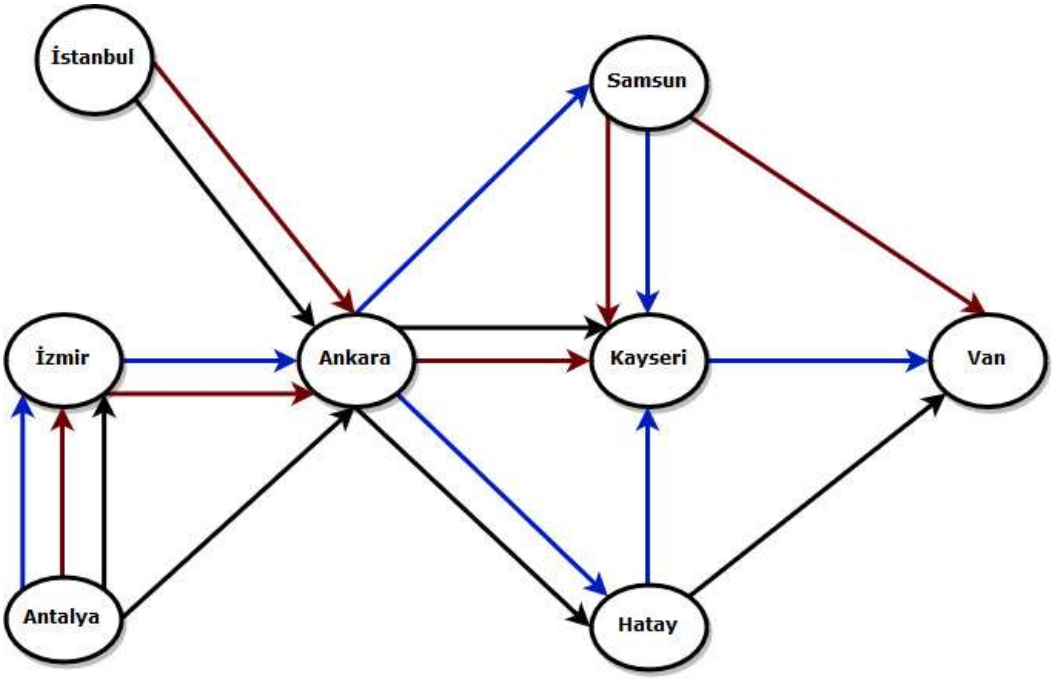

Figure 1: Sample Transportation Network (Red lines: Railway, Blue lines: Airway, Black lines: Highway)

For this sample transportation network input file is structured as Table1. **Please note in your code matrix size should be dynamic. Static arrays will be penalized.**

Table1: A sample input of transportation network.

```
Airway
Istanbul 00000000
Izmir 00100000
Ankara 00000110
Antalya 01000000
Kayseri 00000001
Hatay 00001000
Samsun 00001000
Van 00000000

Highway
Istanbul 00100000
Izmir 00000000
Ankara 00001100
Antalya 01100000
Kayseri 00000000
Hatay 00000001
Samsun 00000000
Van 00000000

Railway
Istanbul 00100000
Izmir 00100000
Ankara 00001000
Antalya 01000000
Kayseri 00000000
Hatay 00000000
Samsun 00001001
Van 00000000
```

- In addition to the input transportation network, you need to read another file named `query.inp` which includes a number of queries about the possible routes on the given network. General format of the query.inp file is as follows:

```
Q1 Istanbul Van 2 H
Q2 Antalya Van Kayseri
Q3 Izmir Van A
Q4 Istanbul Van A1 H1 R1
PRINTGRAPH
```

- In this experiment, you have four different query types. Your goal in this experiment is to answer these queries. **For each query, your code must return all possible paths.**

  - `Q1 City1 City2 N Type` where `City1` and `City2` denotes the source and destination cities respectively and `N` a scalar. Your task here is to find all the paths which use at least N times the specific transportation type given in the query. The specific transportation type could be `H` for Highway, `A` for Airway `R` for Railway. For this query, you need to print a path satisfying the provided conditions, if there is any such path.

  For example, `Q1 Istanbul Van 2 H` means that the source city is `Istanbul`; the destination is `Van`, and you must use at least 2 highways to reach `Van`. The sample output for the given sample network in Figure 1 could be:

```
Istanbul H Ankara A Hatay H Van
Istanbul H Ankara H Kayseri A Van
```

- `Q2 City1 City2 City3` where `City1, City2` and `City3` denotes the source, destination and intermediate cities, respectively. Your task here is to list all the possible paths from `City1` to `City2` by passing through an intermediate city denoted here as `City3` only once.

  For example, `Q2 Antalya Van Kayseri` means the source city is `Antalya`, destination city is `Van` and the intermediate city is `Kayseri`. You need to list the paths from `Antalya` to `Van` by passing through `Kayseri`.

```
Antalya A Izmir A Ankara A Hatay A Kayseri A Van
Antalya A Izmir A Ankara A Samsun A Kayseri A Van
Antalya A Izmir A Ankara H Kayseri A Van
Antalya H Ankara A Hatay A Kayseri A Van
Antalya H Ankara A Samsun A Kayseri A Van
Antalya H Ankara H Kayseri A Van
```

- `Q3 City1 City2 Type` where `City1` and `City2` denotes the source and destination cities respectively and `Type` can be `H` for Highway, `A` for Airway `R` for Railway.

  For this query, your task here is to list the paths from `City1` to `City2` by going over the specific transportation type represented by `Type`, if any such path exists.

  For example, `Q3 Izmir Van A` means that the source city is `Izmir`, the destination is `Van`, and your task here is to list all possible paths while you are going from `Izmir` to `Van` by using only railways. The sample output for the given sample network in Figure 1 could be:

```
Izmir A Ankara A Hatay A Kayseri A Van
Izmir A Ankara A Samsun A Kayseri A Van
```

- `Q4 City1 City2 A{a} H{h} R{r}` where `A`, `H` and `R` denotes the transportation types airway, highway and railway, respectively. The parameter, `a`, `h`, and `r` represent scalar values that are either 0 or positive and that represent how many times a specific transportation type should be used. Your task here is to find all possible paths from `City1` to `City2` satisfying the given constraint.

For example, `Q4 Antalya Van A3 H1 R0` means that the source city is `Antalya`; the destination is `Van`, and you must use 3 airways route, 0 railway route, 1 highway route to reach `Van`. It is not important order of the type of transportation (A3-H1-R0). The sample output for the given sample network in Figure 1 could be:

```
Antalya A Izmir A Ankara A Hatay H Van
Antalya A Izmir A Ankara H Kayseri A Van
Antalya H Ankara A Hatay A Kayseri A Van
Antalya H Ankara A Samsun A Kayseri A Van
```

- `PRINTGRAPH` this query will print graph structure. The sample output for the given sample network in Figure 1 could be:

```
Istanbul --> Ankara
Izmir --> Ankara
Ankara --> Hatay Samsun Kayseri
Antalya --> Izmir Ankara
Kayseri --> Van
Hatay --> Kayseri Van
Samsun --> Kayseri Van
Van -->
```

Grading and Evaluation

- Your work will be graded over a maximum of 100 points.
- There are five query types for this experiment and the contribution of each query to your total score will be partial according to the grading policy stated below.

Q1	20
Q2	25
Q3	20
Q4	25p
PRINTGRAPH	10p

- Your code will be tested with transportation networks which have different complexities. And one output file will be expected as output file that consists the result of your queries from each testing.
- As shown above, it is not fixed of the number of queries in the query file. So you should execute each line separately.
- You have to combine all transportation types in one graph. Three different graph for each transportation type will be penalized.
- Your programs are going to be tested on the dev machine.
- After processing each query, the results are going to be written to the output file which is called **result.out**. Each query's output is going to contain the query and related result. General format of the output file is as follows:

```
Q1 Istanbul Van 2 H
Istanbul H Ankara A Hatay H Van
Istanbul H Ankara H Kayseri A Van
Q2 Antalya Van Kayseri
Antalya A Izmir A Ankara A Hatay A Kayseri A Van
Antalya A Izmir A Ankara A Samsun A Kayseri A Van
Antalya A Izmir A Ankara H Kayseri A Van
Antalya H Ankara A Hatay A Kayseri A Van
Antalya H Ankara A Samsun A Kayseri A Van
Antalya H Ankara H Kayseri A Van
Q3 Izmir Van A
Izmir A Ankara A Hatay A Kayseri A Van
Izmir A Ankara A Samsun A Kayseri A Van
Q4 Antalya Van A3 H1 R0
Antalya A Izmir A Ankara A Hatay H Van
Antalya A Izmir A Ankara H Kayseri A Van
Antalya H Ankara A Hatay A Kayseri A Van
Antalya H Ankara A Samsun A Kayseri A Van
PRINTGRAPH
Istanbul --> Ankara
Izmir --> Ankara
Ankara --> Hatay Samsun Kayseri
Antalya --> Izmir Ankara
Kayseri --> Van
Hatay --> Kayseri Van
Samsun --> Kayseri Van
Van -->
```

Submit Format

Assignment3.zip/ (*Required*)

src/;(*Required*)

src/Makefile; (*Required*)

src/main.java; (*Required*)

src/*.java(optional)

- The program will be executed with three command line arguments:
  <transportation network input file> <query file> <output file>

  Usage example:

```
>javac main.java
>java main transportation_network.inp query.inp result.out
```

  NOTES AND RESTRICTIONS:

- Your experiment should be submitted before the due date. Late submissions will be penalized.
- Use Eclipse while development.
- All assignments must be done individually unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudo code) will not be tolerated.
  In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.