PROJECT: **5**
DUE DATE: **December 9, 2022**

Description:

Implement a graph data structure for a practical application.

Input: Two files - one for the airport information and the other for the connection data.

• airports.csv – IATA *3-letter airport code*,city,airport name,state
• distances.csv – from IATA,to IATA,distance

Output: A menu driven system which has the following options:

1. Read the original data files and store the data to appropriate data structures.
2. Let the user of this program enter an airport code and your program should print out the airport information.
1. Find the connection between two airports. Get two airport codes and find the shortest distance between two airports.
3. *Insert a connection (edge) between two airports -- the user will be asked to enter the two airport codes and its distance. Note that if a pair of airport codes already exists or if the airport code doesn't exist, print out a warning message.*
4. *Delete a connection (edge) -- the user will be asked to enter two airport codes for this connection. Note that if the connection entered doesn't exist, print out a warning message.*
5. Exit.

Create a Java Digraph class to store the airport and distance information. Use Dijkstra's algorithm discussed in class for finding the shortest distances between airports. In your project report, you will need to discuss the ADTs and data structures used for the graph and Dijkstra's algorithm. You will also need to analyze the time complexity of your program with your selected data structures.

Required I/O:

```
Airports v0.1 by F. Last

Command? H
Q Query the airport information by entering the airport code.
D Find the minimum distance between two airports.
E Exit.

Command? Q LAX BOS XYZ …
LAX - Los Ang…
BOS - Boston…
XYZ - unknown

Command? D LAX IAH
Los… to George… is 1576 through the route:
LAX – Los Ang…
…
IAH -

Command? E
```

Possible responses:

Invalid Command
Airport code unknown
Airports not connected

```
Command? I LAX BOS 2999
Los… to Bos… with a distance of 2999 added.

Command? R LAX BOS
Los… and Bos… removed.
```

Project report:

- Page 1: Cover page with your name, class, project, and due date
- Page 2 to 3:
    - Section 1 (**Project specification**): Your ADT description. Description of data structures used and a description of how you implement the ADT
    - Section 2 (**Testing methodology**): Description of how you test your ADT, refer to your testing output. Explain why your test cases are rigorous and complete. Demonstrate that you test each method.
    - Section 3 (**Lessons learned**): Any other information you wish to include.

Turn in:

1. Project report submitted via Canvas.
2. Submission include ONLY Java source files: *AirportApp.java* and all of the supporting classes. NO airports.csv and distances.csv.

```
zip p5 AirportApp.java ADTs.java
scp p5.zip bronconame@login.cpp.edu:/user/tvnguyen7/cs2400-00x/bronconame-p5.zip
```

NO package. You should check out your project on the CPP intranet using:

```
unzip p5
javac AirportApp.java
java AirportApp
```

Grading Guide:

- 10%: Project report and project output.
- 80%: Program correctness. If your program does not compile, you may get partial or zero points here. You must develop your own hash table and hash functions, otherwise, you will get zero here.
- 10%: Coding – efficiency, style, comments, formats

Notes:

1. The following information is required in the beginning of every source file.

```
//
//    Name:         Last, First
//    Project:      #
//    Due:          date
//    Course:       cs-2400-0x-f22
//
//    Description:
//                  A brief description of the project.
//
```

2. The submission **must** be legibly printed.