EE 360C — Algorithms — Summer 2013
Programming Assignment #2

Due: July 17, 2013 11:59pm (via Blackboard)

Programming assignments are to be done individually. You may discuss the problem and general concepts with other students, but there should be no sharing of code. You may not submit code other than that which you write yourself or is provided with the assignment. This restriction specifically prohibits downloading code from the Internet.

Problem Description

Mr. Edision has got tremendous business success by adopting the Stable Matching algorithm and he is aiming to create day care centers across the nation. To do this, He needs to fly across the nation to do marketing campaigns and he is competing with time as other competitors might realize his secret weapon and saturate the market. Your goal is to find a round trip route (i.e., sequence of flights) from a given airport at a provided start time to a desired destination airport and back in the shortest amount of time, given a series of constraints.

Mr. Edision will give (1) a departure airport, (2) a start time when he can be at the departure airport, (3) a destination airport, and (4) the time, in hours, required at the destination before returning (e.g., if Mr. Edision needs to stay for two days, he will provide this information as 48 hours). The task is to find a sequence of flights that will get him to the destination airport at the earliest time and back to the departure airport as early as possible *with the following restrictions*:

- the flight plan must allow a minimum layover of one hour at any intermediate airport;
- the initial flight out of the departure airport must be at least two hours after the stated start time to account for check-in and security procedures; and
- the return flight out of the destination airport must allow him the stated amount of time in the destination city *plus* an additional two hours for check in and security at the destination airport.

Your job is to help Mr. Edision by devising and implementing a variation of Dijkstra's shortest path algorithm. The algorithm shall run in $O(m \log n)$ time where n is the number of airports and m is the number of flights.

Thoroughly read this document, *especially* regarding input and output format before beginning to avoid getting a zero for non-comformance.

Input and Output Specification

Here's some sample flight schedule from Mr. Edision:

```
BOS HOU 1000 A 22
```

It means that the source airport code is BOS, the destination airport code is HOU, he wants to start at 10:00 AM and needs 22 hours at his destination. The output result shall be:

```
TW 53 (BOS 1203 PM --> STL 223 PM)
WN 759 (STL 400 PM --> HOU 545 PM)
WN 590 (HOU 605 PM --> LAX 825 PM)
UA 28 (LAX 955 PM --> JFK 549 AM)
TW 44 (JFK 650 AM --> BOS 754 AM)
```

The first two output lines are for Outbound trip Itinerary, which mean that lapsed time for outbound trip (BOS to HOU), including check in and layover(s) is 8 hours and 45 minutes. The last three output lines are for Return trip Itinerary, which mean that elapsed time for return trip (HOU to BOS), including check in and layover(s) is 15 hours and 9 minutes. Your output should strictly follow the example.

The files `airport-data.txt` and `flight-data.txt` contain airline schedules (with over 3500 flights) from 1992 collected by Roberto Tamassia from EasySABRE. The file `airport-data.txt` starts with the number of cities (an integer). Then for each city there is a line with two items (separated by a tab). The first is a 3 letter airport code. The second is an offset from Greenwich mean time (GMT).

The file `flight-data.txt` contains one line per flight that is tab delimited with the following 8 items: airline, flight number, code for source airport, local departure time, A or P (for am or pm) for departure time, code for destination airport, local arrival time, A or P for arrival time.

Implementation Hints

- Use an adjacency list representation of the directed weighted multigraph defined by the flights among the airports.
- You'll need to create a min-heap data structure to implement Dijkstra's algorithm.
- The times given for the flights are *local* times for the given airport. You have to convert all times to GMT by using GMT offset data in `airport-data.txt`. It will simplify your code if you convert times to minutes into the day. For example 2:30AM would be 150 minutes into the day.
- To test your code, it would be helpful to make a smaller version of the airport and flights data.

Submission Instructions

- Make sure your program compiles on LRC machines before you submit it. It should compile using the standard commands `javac *.java`, `gcc *.c`, or `g++ *.cc` without any extra switches or any additional libraries.
- You should submit a single zip file titled `eid_lastname_firstname.zip` that contains all your program files and optionally a readme file. Do not put these files in a folder before you zip them (i.e. the files should be in the root of the ZIP archive) and do not include binaries.
- Your solution must be submitted via Blackboard *before* the deadline. No late submissions will be accepted.