CSC30500 Project #1
DUE: Tuesday, September 16, 2014, 11:59 PM

1 Objectives

- Writing a file based database system
- Becoming familiar with the needs for a true database system

2 Problem Statement

You will be writing a computer program that keeps track of cities, airline names, flights, and costs. Your program should essentially prompt the user to enter a one letter command, and process the command as follows:

- **a** (for add), which should then read in one of:
  - **c**, which should then also read the next string as a city code and then the reminder of the current input line as a city name. For example, the entire command:

    ```
    a c stl saint louis
    ```

    should add the city of `saint louis` to the list of available cites, with a city code of `stl`.
  - **a**, which should then read the next string as an airline abbreviation and then the remainder of the line as an airline name. For example, the command:

    ```
    a a twa trans world airlines
    ```

    should add an airline named `trans world airlines`, with a code of `twa`.
  - **f**, which should then read the next string as an airline abbreviation, the next string as a departure city, another string as a destination city, and finally a positive integer as the cost (in dollars) of the flight. For example, the command:

    ```
    a f twa stl jfk 119
    ```

    should add a flight from `stl` to `jfk` that costs `$119` on the airline whose code is `twa`.
- **l** (that is, the letter 'l' for "list"), which should then read in one of:
  - **c**, which should list *all* of the city names and their corresponding codes.
  - **a**, which should list *all* of the airline names and their corresponding codes.
  - **f**, which should list *all* of the flights, including (for each flight) the airline *name*, the departure city *name*, the arrival city *name*, and the cost of the flight.
- **f** (for "find"), which should also read in a departure airport code, an arrival airport code (in that order), and finally either the number 0 or the number 1 (representing the number of connections allowed during the trip). The program should then print out all of the matching trips matching the number of connections requested (i.e. 0 would correspond to a direct flight, while 1 would correspond to two flights with the second flight departing from the airport that the first one arrives in). For each matching flight, the output should have the departing airport code, arrival airport code, airline code, and total cost. An example of this command might be:

  ```
  f stl jfk 0
  ```

  or

```
f stl jfk 1
```

Note that it is possible that such requests might generate no results; a message should probably be printed indicating such.

- **q** (for "quit"), which should stop the running program.

Some notes:

- After processing one command, the program should continue to prompt the user for another command (and process what the user enters). This repeats until the user enters 'q'.
- *Data should be persistent between runs!* That is, if you add the city of saint louis to the list of cities on the first run and then quit, then when you run the program again (even if this happens decades later), the city of saint louis should still show up if you list the cities as your first command.

3 Example Execution

Suppose the first time you run the program, you do the following (`>>>` is the program's prompt to the user):

```
>>> a c stl saint louis
>>> a c alb albany
>>> a c jfk new york
>>> a c ord chicago
>>> a a twa trans world airlines
>>> a a aa american airlines
>>> a f twa stl ord 90
>>> a f aa stl jfk 119
>>> a f aa ord jfk 97
>>> a f twa stl jfk 109
>>> a f twa jfk alb 49
>>> l c
stl saint louis
alb albany
jfk new york
ord chicago
>>> l a
twa trans world airlines
aa  american airlines
>>> l f
trans world airlines: saint louis -> chicago $90
american airlines: saint louis -> new york $119
american airlines: chicago -> new york $97
trans world airlines: saint louis -> new york $109
trans world airlines: new york -> albany $49
>>> f stl jfk 0
stl -> jfk : aa $119
stl -> jfk : twa $109
>>> f stl jfk 1
stl -> ord : twa $90; ord -> jfk : aa $97, for a total cost of $187
>>>q
```

Then, suppose you run the program again (make note that this demonstrates *one* facet of the persistence of the data you just entered:

```
>>> l c
stl saint louis
alb albany
jfk new york
ord chicago
>>> a a sw southwest airlines
>>> l a
twa trans world airlines
aa  american airlines
sw southwest airlines
>>> q
```

Make careful note that stl, jfk, and ord were *not* explicitly entered during the second run, but they should still show up, as they were added during the first run!

4 What To Hand In

You will be submitting your source code and a `read.me` file via blackboard. The `read.me` file should include information about your project including (but not limited to):

- your name
- the date
- the platform you developed your code on (Windows, Linux, ...)
- any special steps needed to compile your project
- any bugs your program has
- a brief summary of how you approached the problem

You might also want to consider adding things like a "software engineering log" or anything else you utilized in getting the project done.

5 Grading Breakdown

Code Compiles	30%
Following Directions	20%
Correct Execution	40%
Code Formatting/Comments	10%
Early Submission Bonus	*5%*

6 Notes & Warnings

- This is *not* the kind of project you will be able to start the night before it is due - instead, *START NOW!!!*
- Projects may *not* be worked on in groups or be copied from *anyone*. Failure to abide by this policy will result in disciplinary action(s). See the course syllabus for details.
- Have you started working on this project yet? If not, then *START NOW !!!!!*