CSC30500 Project #1
DUE: Thursday, September 14, 2017, 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, baseball teams, and games played between baseball teams. 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`.
  - **t**, which should then read the next string as a city code (as in reading in a city above), and then another string representing a team name (which will not have whitespace in it). For example, the command:

    ```
    a t stl Cardinals
    ```

    should add a team named `Cardinals`, with a city code of `stl`.
  - **g**, which should then read in four items, as follows:
    - a "visiting" team name as a (non-whitespace) string,
    - a "visiting" team score as a (positive) integer,
    - a "home" team name as a (non-whitespace) string,
    - a "home" team score as a (positive) integer.

    For example, the following command:

    ```
    a g Cubs 3 Cardinals 5
    ```

    should add a home game for the Cardinals, with the Cubs being the visiting team. The Cardinals won this game by a score of 5 to 3.
- **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.
  - **t**, which should list *all* of the team names and their corresponding city *name*. Note that you should print the city name, *not the city code*.
  - **g**, which should list *all* of the games, including (for each game) the visiting team *name*, the visiting team score, the home team name, and the home team score.
- **r** (for "record"), which should also read in a team name. This command should result in printing the team's total number of wins, total number of losses, total points (runs, in baseball parlance) scored, and total points (runs, in baseball parlance) scored against them. Examples of running this command might be:

```
r Cardinals
```

or

```
r Cubs
```

Note that it is possible that such requests might be for a team that does not exist; a message should probably be printed indicating such.

- **s** (for "standings"), which should print out the record (see the previous command) for each team in the system. The teams should be printed in the order specified by the following scheme:
  - teams should be printed from highest winning percentage to lowest winning percentage. Winning percentage can be calculated as number of wins divided by total number of games played, noting that total number of games played is number of wins plus number of losses. Note that baseball games cannot end in ties.
  - teams with equal winning percentages should be sorted by point (run, in baseball parlance) differentials, high to low. Point differential is calculated as total points scored by the team minus total points scored against the team.
  - teams with equal winning percentages and equal point differentials should be sorted by their team name (low to high).
- **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 chi Chicago
>>>a c phi Philadelphia
>>>l c
stl Saint Louis
chi Chicago
phi Philadelphia
>>>a t stl Cardinals
>>>a t phi Phillies
>>>a t chi WhiteSox
>>>a t chi Cubs
```

```
>>>l t
Saint Louis Cardinals
Philadelphia Phillies
Chicago WhiteSox
Chicago Cubs
>>>a g Cubs 4 Cardinals 5
>>>a g Cubs 1 Cardinals 8
>>>a g Phillies 2 Cubs 1
>>>a g Phillies 0 WhiteSox 9
>>>a g WhiteSox 2 Cubs 3
>>>a g Cubs 7 WhiteSox 11
>>>a g Cardinals 3 Phillies 1
>>>a g Cardinals 2 Phillies 6
>>>l g
Cubs                4      Cardinals           5
Cubs                1      Cardinals           8
Phillies            2      Cubs                1
Phillies            0      WhiteSox            9
WhiteSox            2      Cubs                3
Cubs                7      WhiteSox            11
Cardinals           3      Phillies            1
Cardinals           2      Phillies            6
>>>r Cubs
====Team==================================W===L===PF===PA
Cubs                                      1   4   16   28
>>>r Cardinals
====Team==================================W===L===PF===PA
Cardinals                                 3   1   18   12
>>>s
====Team==================================W===L===PF===PA
Cardinals                                 3   1   18   12
WhiteSox                                  2   1   22   10
Phillies                                  2   2    9   15
Cubs                                      1   4   16   28
>>>q
```

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

```
>>>l t
Saint Louis Cardinals
Philadelphia Phillies
Chicago WhiteSox
Chicago Cubs
>>>s
====Team====================================W===L===PF===PA
Cardinals                                   3   1   18   12
WhiteSox                                    2   1   22   10
Phillies                                    2   2    9   15
Cubs                                        1   4   16   28
>>>a c oak Oakland
>>>a t oak A's
>>>l t
Saint Louis Cardinals
Philadelphia Phillies
Chicago WhiteSox
Chicago Cubs
Oakland A's
>>>a g WhiteSox 4 A's 6
>>>l g
Cubs                4      Cardinals           5
Cubs                1      Cardinals           8
Phillies            2      Cubs                1
Phillies            0      WhiteSox            9
WhiteSox            2      Cubs                3
Cubs                7      WhiteSox            11
Cardinals           3      Phillies            1
Cardinals           2      Phillies            6
WhiteSox            4      A's                 6
>>>s
====Team====================================W===L===PF===PA
A's                                         1   0    6    4
Cardinals                                   3   1   18   12
WhiteSox                                    2   2   26   16
Phillies                                    2   2    9   15
Cubs                                        1   4   16   28
>>>q
```

Make careful note that cities, teams, and games are already populated with data when this run occurred, despite those values not being explicitly entered during this run!!!

4 What To Hand In

You will be submitting your source code and a `read.me` file within a zip file via Canvas. 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	20%
Following Directions	20%
Correct Execution	40%
Code Formatting/Comments/`read.me`	20%
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 !!!!!*