Lab08: Design and Simulation Lab Due: Jan 22, 2019 at 3:30pm

The goal of this lab assignment is for you to become proficient with the concept and execution of a program that runs a simulation. Specifically, we will target a baseball game given the statistics of the players involved and run our simulation from there. We will use the most recent World Series involving the Dodgers and Red Sox as an example.

Inputs: You will be given two input files that has the statistics of the starting lineups for both the Dodgers and Red Sox (Dodgers.tsv and RedSox.tsv). The format for each of these files will be as follows (the first line is NOT included in the files but I've added them here so you know what each column represent):

Last Name	First Name	Batting Average	Hits	Doubles	Triples	Home Runs
Dozier	Brian	.182	26	9	0	5
Turner	Justin	.312	114	31	1	14

Note that each entry in a row is separated by a tab (which is represented in Python as `\t`.)Your program will prompt the user for how many World Series they wish to simulate between the two teams. Please note the there will be no GUI interface for this program, as all interaction will occur in the Python shell.

Outputs: Depending upon the number of simulations the user enters, your output will vary. If the user enters "1", then you will run a single World Series and display the sample results as follows:

```
Results of World Series Simulation

Game 1: Red Sox 6 Dodgers 4
Game 2: Dodgers 3 Red Sox 1
Game 3: Red Sox 8 Dodgers 3
Game 4: Red Sox 3 Dodgers 0
Game 5: Red Sox 5 Dodgers 3

Red Sox win the series 4-1

Home Runs:
Dodgers - B. Dozier 2, D. Freese 3, E. Hernandez 3
Red Sox - M. Betts 3, A. Benintendi 2, S. Pearce 2, J. Martinez 1
Runs scored leader(s):
M. Betts (Red Sox) 7
```

If the user entered more than one for the number of simulations to run, the shell output should be something like:

```
Results of 1000 World Series Simulations

Dodgers win in 4 - 4.7%
Dodgers win in 5 - 10.7%
```

```
Dodgers win in 6 - 14.1%
Dodgers win in 7 - 15.6%
Red Sox win in 4 - 8.0%
Red Sox win in 5 - 12.9%
Red Sox win in 6 - 17.5%
Red Sox win in 7 - 16.5%
```

(Note: it's a good idea to check that your percentages sum to 100%).

In addition you will produce an output file that varies depending on the number of simulations run. If only one simulation, the output file should be called WSplaybyplay.py and it should contain the play-by-play results. For example:

```
Dodgers-Red Sox World Series Simulation

========== Game 1 ==========

Inning 1 - Dodgers
B. Dozier was out
J. Turner singled
D. Freese homered (J. Turner scored)
M. Machado was out
C. Taylor was out

Inning 1 - Red Sox
M. Betts singled
A. Benintendi homered (M. Betts scored)
S. Pearce was out
J. Martinez was out
X. Bogaerts singled
R. Devers was out

Score: Dodgers 2 Red Sox 2

Inning 2 - Dodgers
M. Kemp was out
E. Hernandez homered
Y. Puig was out                          [etc.]
```

If the user asked for multiple simulations, the output file should be called WSsimulations.py and should simply look like:

```
Dodgers-Red Sox World Series Simulation

1: Dodgers win in 5
2: Dodgers win in 4
3: Red Sox win in 6
4: Dodgers win in 6                          [etc.]
```

We will assume the following for this simulation:

- Each player's batting average represents the percentage of time they safely get a hit (i.e. NOT out). The hits given represent the total number of singles, doubles, triples, and home runs so you should be able to figure out what percentage of "hits" are doubles, etc.
- Each team gets 3 outs per inning and, to make things simple, runners only advance as far as the batter advances (i.e. if a runner is on second, they will go to third base on a single and score on a double or higher). We are not taking pitching/fielding into account for the purposes of this simulation.
- Each game is nine innings unless it is tied in which case you must play extra innings until the tie is broken. A full final inning will be played even if the second team at bat has already won.
- The World Series is best of 7 games and should stop as soon as one team has won 4 games.

Implementation: I am expecting two modules from you when finished: WSSimulation.py and WSPlayer.py. The latter is a class containing ALL pertinent data for each instance of a player. The first file should contain all of the functions you need to implement this simulation. Note that I will be looking for evidence of your design process in this file (i.e. Top-Down Design or prototyping). You should also include a brief description at the top of the module explaining which process you did and why. Finally, please follow all formatting shown in the examples.

Grading: This lab is worth 10 points and will be graded as follows:

- 4 points for successful implementation of single series simulation
- 3 points for successful multiple series implementation
- 1 points for adhering to the 100 line limit on functions including comments
- 1 point for adhering to proper encapsulation techniques for the WSPlayer class
- 1 points for thorough comments including your note about your design strategy

As this program is due after the end of the semester, it will *not* be a part of your first semester grade.