CS 260 Programming Assignment #4
Due 12/5/14
A Ranking System for College Football Teams
Using a Weighted Directed Graph

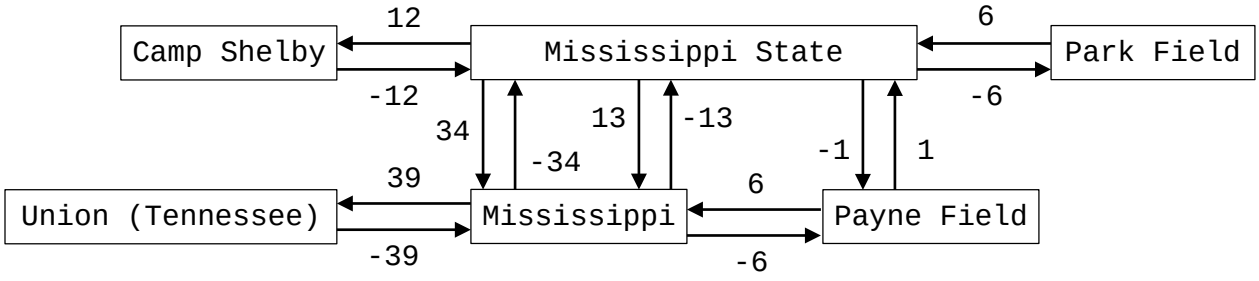
Write a Python program that implements a rudimentary system for ranking college football teams, as follows. The system is unbiased but naïve, because it only considers final scores of games played.

First read a file that contains scores of football games, and store the information as a weighted directed graph G using an adjacency lists data structure. Each team will be represented by a vertex, and each game will be represented by two directed edges. If teams X and Y play a game in which team X scores A points and team Y scores B points, then the graph should include an edge X→Y with weight w(X,Y) = A–B, and also an opposite edge Y→X with weight w(Y,X) = B–A.

Here is an example input file, which has actual scores from the 1918 season (south region).
We use this file to illustrate, because it is short enough to trace through the computations.
Several additional (much larger) example input and output files are also available on Blackboard.

```
Payne Field                7 Mississippi State        6
Camp Shelby                0 Mississippi State       12
Payne Field                6 Mississippi             0
Park Field                 6 Mississippi State       0
Union (Tennessee)          0 Mississippi             39
Mississippi                0 Mississippi State       34
Mississippi State         13 Mississippi             0
```

Here is the weighted directed graph G that should be constructed from the above input data:



Next your program will compute a numerical rating for each team using the following algorithm:

For each team X, define performance[X] = average weight of all edges that exit from vertex X.
For each team X, initialize rating[X] = performance[X].
Repeat 5000 times:
    For each team X, compute schedule_factor[X] = average of rating[Y] over all edges X→Y.
    For each team X, update rating[X] = performance[X] + schedule_factor[X].

Intuitively, the algorithm starts with no information about the strength of each team's opponents. But as more information is learned about each team's opponents, this affects the team's schedule_factor, which in turn affects the team's rating, which affects its opponents' ratings, etc. So the information and calculations gradually propagate throughout the entire graph until all the teams' ratings converge to an equilibrium.

Continuing with the previous example, here we illustrate some of the algorithm's calculations in the table below. All the team rating values in this short example converge to steady state in fewer than 100 iterations, but other larger files may need thousands of iterations to reach equilibrium.

	initially		1st iteration		2nd iteration		...	5000th iteration	
Team	perf	rating	sched	rating	sched	rating	...	sched	rating
Camp Shelby	-12	-12	10.4	-1.6	8.5	-3.5	...	11.2	-0.8
Mississippi	-3.5	-3.5	-3.675	-7.175	-4.6375	-8.1375	...	-5.1	-8.6
Mississippi State	10.4	10.4	-1.9	8.5	1.48	11.88	...	0.8	11.2
Park Field	6	6	10.4	16.4	8.5	14.5	...	11.2	17.2
Payne Field	3.5	3.5	3.45	6.95	0.6625	4.1625	...	1.3	4.8
Union (Tennessee)	-39	-39	-3.5	-42.5	-7.175	-46.175	...	-8.6	-47.6

Finally, your program will output all the teams sorted in descending order by their final rating values. Display each team's rank, team name, number of games won/lost/tied, and final rating value, using the following format.

```
Rank Team                          W-L-T    Rating
  1  Park Field                    1-0-0      17.2
  2  Mississippi State             3-2-0      11.2
  3  Payne Field                   2-0-0       4.8
  4  Camp Shelby                   0-1-0      -0.8
  5  Mississippi                   1-3-0      -8.6
  6  Union (Tennessee)             0-1-0     -47.6
```

Input/output specifications:

- Your program should run using the command "python programname.py filename.txt" where filename.txt denotes the input file name. For example, "python myprogram.py 2013.txt". Also, we are using Python 3, so on some computers you might need to type python3 rather than just python.
- Your program should prepend the characters "output_" to the beginning of the input file name to obtain the output file name. For example, if the input file is "2013.txt", then the output file should be named "output_2013.txt".
- Input and output formats must be exactly as shown in the example files that are available on Blackboard. Do not change these input or output formats, or you may receive a failing grade if we are unable to easily test your program due to using unexpected file formats.
- You may assume that the number of teams ≤ 1000, and the total number of games ≤ 6000. Also, the length of each team name ≤ 32 characters, or exactly 32 characters including trailing blanks.