Assignment 3
Number Crunching
Due: Monday, April 4, 2014, 11:59pm

It's mid-March. Spring is in the air. The grass is getting greener and the flowers are starting to bloom. And you know what that means: it's time for March Madness!

Contrary to what you might think, March Madness is not a mental disorder. It is actually a nickname for a basketball tournament held in March and early April. The participants of this tournament are the top 68 college/university basketball teams in the United States. These top 68 teams are selected by a special committee, which takes into account various statistics including teams' win-loss records, points scored, points allowed and strength of schedule in the regular season.

Why strength of schedule? The reason is that there are over 350 university basketball teams competing across the US. So each team cannot play against all other teams. Instead, each team schedules games against only about 30-35 other teams. As a result, some teams play good teams, while others play terrible teams. This leads to a problem: if team A wins a lot against a bunch of bad teams, it does not necessarily mean that team A is a good team. Similarly, if team B plays a lot of good teams and loses a few games, it doesn't mean team B is a worse team.

To deal with this problem, one tool that is often used is RPI (Ratings Percentage Index). The higher the RPI of a team, the better that team is considered. A basketball team's RPI is given by the simple formula:

RPI = 0.25 * Team's Winning Percentage + 0.5 * Opponent's Winning Percentage
+ 0.25 * Opponent's opponent's winning percentage

Obviously, this is a flawed formula, but it gives some measure of how well a team performed corresponding to its strength of schedule.

In this assignment, your job is to do some basic number crunching in the area of sports statistics. You are provided a bunch of data in CSV (comma-separated values) files, and your job is to read that data, compute RPI and some other numbers, and be able to answer queries related to the data.

Why basketball data? Because this was the easiest sports dataset for me to find that was in a relatively simple format. Secondly, there is a competition going on for $15,000 for whoever can best predict the winner of March Madness (http://www.kaggle.com/c/march-machine-learning-mania). So there is a great deal of interest in analyzing data in this domain. Thirdly, you will find that in every domain where data analysis is required, CSV files are often the preferred input format. So being able to read in, parse and extract useful information from CSV files is a very useful skill.

Input Format:

The input files can be found at http://www.tahirazim.com/cs212/assignments/assignment3/csvs/. The format of the input files is described below:

teams.csv

This file identifies the 356 different college teams that are present in at least one of the seasons from 1995-1996 through 2013-2014. The other data files that identify teams, such as when game-by-game results are listed or tournament seeds are listed, will always reference the teams by their id number rather than by their name. This

makes the files more compact and also eliminates any possible spelling issues. This is the only file that actually contains the text names of the college teams.

"id" - this number uniquely identifies the college team. All id values are three digit numbers, starting with 501, and the id's are assigned in alphabetical order, so for instance 501 is "Abilene Chr", and 502 is "Air Force", ... and 855 is "Youngstown State", the last team name alphabetically. The one exception to the alphabetical sequencing is Incarnate Word (#856), which was added recently. Having numbers so high avoids any possible confusion with game scores and ensures that all team id's will be exactly three digits long.

seasons.csv

This file identifies the 18 different seasons included in the historical data.

"season" - indicates the letter used to uniquely identify each season. For instance, season Q represents the 2011-2012 season, meaning the college basketball season that started in late 2011 and ended with the final tournament during March/April 2012.
"years" - indicates the years spanned by each season. For instance, you can see that season Q was played during the years 2011-2012.

regular_season_results.csv

This file identifies the game-by-game results for all 18 seasons of historical data, from season A (1995-6) through season R (2012-3). Each year, it includes all games played from daynum 0 through 132. Each row in the file represents a single game played.

"season" - this is the one-letter identifier of the season, corresponding to the "season" column in the "seasons.csv" file.
"daynum" - this integer always ranges from 0 to 132, and tells you what day the game was played on.
"wteam" - this identifies the id number of the team that won the game, as listed in the "teams.csv" file. No matter whether the game was won by the home team or visiting team, "wteam" always identifies the winning team.
"wscore" - this identifies the number of points scored by the winning team.
"lteam" - this identifies the id number of the team that lost the game.
"lscore" - this identifies the number of points scored by the losing team.
"wloc" - this identifies the "location" of the winning team. If the winning team was the home team, this value will be "H". If the winning team was the visiting team, this value will be "A". If it was played on a neutral court, then this value will be "N".

tourney_results.csv

This file identifies the game-by-game tournament results for all 18 seasons of historical data, from season A (1995-6) through season R (2012-3). The data is formatted exactly like the "regular_season_results" data, except that all games are assumed to be on a neutral court, and therefore the "wloc" column is not included.

regular_tourney_combined_results.csv

This file combines the data from regular_season_results.csv and tourney_results.csv into one file. If you feel more comfortable using just one file of game data, feel free to use only this file. The format is exactly the same as regular_season_results.csv.

Output:

Your job is to support several kinds of queries on the dataset provided above. Following is a list of mandatory queries you need to support:

1. Who was the RPI champion at the end of a given season?

`./madness rpiChampion <SEASON>`

2. Who was the tournament champion in a given season? Note that this is the winner of the final game of the tournament for that season. In other words, the winner of the last entry in the file for that season.
   `./madness champion <SEASON>`
3. What was the list of games played by a team and their results in a given season?
   `./madness list <SEASON> <TEAM>`
4. What was the final record (wins and losses) of a team in a given season?
   `./madness finalRecord <SEASON> <TEAM>`
5. Which team won the most games in a given season?
   `./madness mostWins <SEASON>`
6. Which team had the most away wins in a given season?
   `./madness mostAwayWins <SEASON>`
7. Which team had the best win percentage in a given season?
   `./madness bestWinPercentage <SEASON>`
8. Which team scored the most points in a given season?
   `./madness mostPoints <SEASON>`
9. Which team allowed the fewest points in a given season?
   `./madness fewestPointsAllowed <SEASON>`
10. Which team had the highest average margin of victory in a given season?
    `./madness largestMargin <SEASON>`

Note that the end of a season means the end of both the regular season and the tournament. Also, if there is a tie between two teams for a given query, then either team's name is acceptable in the output. For example, if two teams won the same number of games in a season, then either team can be in the output for query # 5. Finally, in your final submission, you should not print anything else other than what's required. It is likely I will once again test your output using an automated script, so differences in output could lead to deduction in points.

Example Interactions:

```
$ ./madness rpiChampion 2003-2004
Duke
$ ./madness champion 2003-2004
Connecticut
$ ./madness list 1999-2000 Cincinnati
W Youngstown St 94-67
W Cleveland St 91-56
W Santa Clara 88-67
W Iowa St 75-60
W Gonzaga 75-68
W North Carolina 77-68
W MS Valley St 74-48
W St Louis 79-64
L Xavier 64-66
W Oklahoma 72-57
W WI Milwaukee 93-60
W Boise St 78-46
W UNLV 106-66
W Charlotte 81-54
W Marquette 67-48
W Tulane 72-59
```

```
W Ohio 73-59
W Memphis 75-55
W Marquette 72-60
W Louisville 75-65
W South Florida 89-72
W Charlotte 70-62
W UAB 93-80
W DePaul 87-64
W Houston 77-65
L Temple 69-77
W Southern Miss 95-69
W Louisville 68-59
W DePaul 64-62
W St Louis 84-41
L St Louis 58-68
W UNC Wilmington 64-47
L Tulsa 61-69
$ ./madness finalRecord 2003-2004 Duke
31-6
$ ./madness mostWins 2003-2004
Connecticut
$ ./madness mostAwayWins 2003-2004
St Joseph's PA
$ ./madness bestWinPercentage 2003-2004
Stanford
$ ./madness mostPoints 2003-2004
Connecticut
$ ./madness fewestPointsAllowed 2003-2004
Air Force
$ ./madness largestMargin 2003-2004
Gonzaga
```

Programming Hint:

This is an example of an assignment that would be easiest to do if you made extensive use of objects and classes in your implementation. You would probably need at least a *Team* class, a *Season* class, and perhaps a *TeamManager* and a *SeasonManager* class to keep track of the various teams and seasons in the dataset. You are also encouraged to use strings, file streams, and STL containers to complete the assignment.

Helpful Resources:

If you want to compare your results to the actual results, here are some websites that will be useful:

http://statsheet.com/mcb/rankings/RPI?id=1999-2000
http://www.sports-reference.com/cbb/seasons/2004-school-stats.html
http://www.sports-reference.com/cbb/seasons/

Grading Criteria:

80% of the points in this assignment are for functionality. If you give correct answers when your program is run with the above command-line arguments and you don't display any extra output, you will get full points.

20% of the points are style points. This includes decomposing your code into classes and functions, good naming of variables, good indentation, comments, and so on. For style, you will receive either 0, 10 or 20 points:

0 points means your code is in horrible condition.
10 points means your code could use some significant improvements.
20 points means your code is in reasonably good shape.

Bonus Extensions:

There are a number of interesting extensions that can be made in this assignment for extra credit. If you have implemented an extension, you still need to submit by the assignment deadline. However, to get credit for an extension, you need to demo the extension to me in the lab on Friday, April 4. Here are a couple of examples:

i) Think of some new statistics that might be interesting and compute those statistics. You can get 1% extra credit for each new interesting statistic, up to 10% extra credit. Of course, it's a subjective decision as to whether I find the statistic interesting.

ii) 5% extra credit: Find data for some other sport and crunch some similar numbers for that data as well.

iii) Money+publication+fame: Get the full dataset from http://www.kaggle.com/c/march-machine-learning-mania and predict the winner of March Madness for all or almost all of the past 18 years.