CSCI-1200 Computer Science II — Spring 2006
Homework 3 — Tennis Classes

In this assignment you will parse and compute statistics from the results of the most recent *Grand Slam* tennis championship, the 2006 Australian Open. Scoring and determining the winner of a tennis match is quite complex, but we have summarized everything you need to know to complete this assignment. Please read the entire handout before starting to code the assignment.

Tennis Scores

Here's a crash course in tennis scoring: A *match* between two players consists of a number of *sets*. For men's Grand Slam events, the first player to win three sets is the winner of the match (a.k.a., the best of five). Each set in the match consists of a number of *games*. The first player to win six games is the winner of the set, but the player must win by two games. If the set score gets to 6-5 (or 5-6), the players play a twelfth game. If the set score is 7-5 (or 5-7), the set is over. If the players are tied at 6-6 they play a special game called a *tiebreak* and the winner of the tiebreak wins the set, 7-6 (or 6-7). At the Australian Open, there is a special exception and in the fifth set they do not play a tiebreak. They just keep playing regular games until one player is ahead by two games (resulting in some fantastic 5 hour matches!) More details on the tennis scoring system are available here: `http://en.wikipedia.org/wiki/Tennis_score`. As an example of the input you will be parsing, here is the result of the recent men's final:

```
Roger Federer d. Marcos Baghdatis 5-7 7-5 6-0 6-2
```

In this match, player *Roger Federer* defeated player *Marcos Baghdatis* in a four set match. Baghdatis won the first set (it was close). Federer won the second set (it was also close). In the third set Baghdatis didn't win any games and in the last set he only won two games.

File I/O and Command Line Arguments

Your program will run with two command-line arguments, one being the input file containing the match information and the other being the output file where you will write the computed statistics. Example input and output files are posted on the course website. For example, here is a valid command line to your program:

```
tennis_statistics.exe sample_scores.txt sample_scores_out.txt
```

We have provided you with a few sample tennis score datasets. The original data was taken from this online resource: `http://stevegtennis.com/`. The format has been modified slightly to ease parsing. Each match is listed on a separate line. The winner is always listed first. Each player has a first name and a last name (two strings). The special string "`d.`" is placed between the two names. Each set is a string concatenating the number of games the first player won with the number of games the second player won, with the character "`-`" between.

Statistics Collected and Output

The output will be in *three parts*. First is a table with the players ordered by the **percentage of matches** they won. Each row of the table should include the player, the number of matches won, the number of matches lost and the percentage of matches won. Try to do a little bit of nice formatting to this output (see the example code from lecture). For example, given an input file with these matches:

```
Marcos Baghdatis d. Radek Stepanek  6-4 6-3 3-6 0-6 7-5
David Nalbandian d. Danai Udomchoke  6-2 6-2 1-6 6-7 6-1
Marcos Baghdatis d. Ivan Ljubicic  6-4 6-2 4-6 3-6 6-3
Marcos Baghdatis d. David Nalbandian  3-6 5-7 6-3 6-4 6-4
```

Your program will produce a table similar to this:

```
MATCH STATISTICS
Player              W    L   percentage
Marcos Baghdatis    3    0        1.000
David Nalbandian    1    1        0.500
Danai Udomchoke     0    1        0.000
Radek Stepanek      0    1        0.000
Ivan Ljubicic       0    1        0.000
```

The formatting shown above is an example, your output may be formatted differently, as long as it satisfies the description above. The second part of the output is a table with the players ordered by the **percentage of games** they won. As in the first part, each row lists the player, the number of games won, the number of games lost and the percentage of games won. Here's the output for the data above:

```
GAME STATISTICS
Player              W    L   percentage
David Nalbandian   49   44        0.527
Radek Stepanek     24   22        0.522
Marcos Baghdatis   73   69        0.514
Ivan Ljubicic      21   25        0.457
Danai Udomchoke    18   25        0.419
```

The third and final part of the output is a chance for you to be creative. You will collect and output some other statistic from the matches. A simple example would be to output the players ordered by the percentage of sets won. A more complex example would be to find all the matches where the ultimate winner lost the first set and output the players who most often "come from behind to win". Extra credit will be awarded to particularly interesting statistics that require clever programming. The most important task for the this part of the assignment is to write a concise description (< 100 words) of your new statistic. Put this description in a *plaintext* file named `README.txt` along with any other notes for the grader.

Useful Code

To control the formatting of your tables, you'll want to read up on the iomanipulators `std::setw(int)`, `std::setprecision(int)`, and `std::fixed`. And don't forget about the `sort` function that can be used to order the contents of a `vector`. To help you parse a string from the input file that represents a set, we provide the following code to get you started. The `atoi` function takes a C-style string as its argument and returns an integer.

```
// Parse a string that represents a set, i.e., "6-3"
void parse_set(std::string &set, int &games_won, int &games_lost) {
  int i = set.find('-');
  games_won =  atoi(set.substr(0,i).c_str());
  games_lost = atoi(set.substr(i+1,set.size()-i-1).c_str());
}
```

Program Requirements & Submission Details

Your program should involve the definition of *at least one class* that has its own .h and .cpp files, named appropriately. Do all of your work in a new folder named `hw3` inside of your CSII homeworks directory. Put the main function of your code in a file named `main.cpp`. Use good coding style when you design and implement your program. Be sure to make up new test cases and don't forget to comment your code! If you have any notes you want the grader to read, write them in a *plain text* file named `README.txt`. When you are finished please zip up your `hw3` folder exactly as instructed for the previous assignments and submit it through the course webpage.