Fantasy Football Geeks

A lot of people enjoy playing fantasy football. If you are unfamiliar with fantasy sports, the basic idea is that you choose a group of real-life players and are scored each week based on their collective statistics. If the players you chose did well, then your make-believe team gets a win. If the players you chose did poorly, well… better luck next time. The die-hard players pay extra money examine the most complex of statistics. In this field, there are jobs for computer scientists like you. But the field of sports statisticians goes beyond fantasy, many teams hire mathematicians and coders to calculate statistics and help them make roster and playbook decision. This project will help you learn and practice:

- Creating projects with MS Visual Studio
- Designing and implementing software
- Practicing with C# syntax
- Debugging programs
- Working in teams
- Establishing interfaces
- Reading and writing text files
- Parsing and evaluating data

Program Description:

Break up into teams of three. If there are extra students, it's OK to have a couple teams of four. You will be working on a group assignment where each team member completes a different component. If you four people in your group, then two of the team members can work as partners by coding on the same laptop. Just be sure to switch off so that each person gets time at the keyboard. It's probably wise to appoint one person as the "team lead."

You have been given a data file containing game-by-game statistics for 16 NFL quarterbacks over the first five games of the 2019 season. The data file includes the players name, the team he plays for, and then some basic stats for each game (number of pass completions, number of pass attempts, number of yards thrown, and number of touchdowns thrown). The data is stored with pipes (|) separating the major fields and colons (:) separating the statistics within a single game.

Your program needs to calculate the quarterback's overall completion percentage, the total number of yards thrown, and the total number of touchdowns thrown. The program should output the results as a pretty table.

Here are the three tasks that your group will need to complete (one task per-person):

1) **Text File Parsing**: One team member is responsible for reading each of the quarterback's name, team, and game stats from the data file. The data should be read into parallel arrays: the first array is a 16 row by 2 column table of strings that contains the name and team of each quarterback. The remaining data can either be put into a single 16 row by 20 column table containing the stats for each game, or into five tables that each contain 16 rows and 4 columns (e.g., a different table representing each game). It is appropriate to convert all of the statistical numbers to type double (even if the values represent whole numbers, because 4.0 is just as good as 4 for statistical calculations). This might make it easier to store the data.

Note: This role is probably the most difficult.

2) **Computations**: Another team member is responsible for all the statistical calculations. You will receive the parallel arrays that were created by your teammate who parsed the text file. Your job is to run the statistics on each of the 16 quarterbacks to generate the overall statistics: completion percentage, total yards thrown, and total touchdowns thrown.

   Note: This role is probably the easiest; therefore, this person is also responsible for writing the Main function that "glues together" all of the individual components.

3) **User Interface:** The last team member is responsible for outputting the data to the screen. Since your job requires the final statistics, which might not be completed for some time, it makes sense for you to create a bogus, hardcoded table of results for designing your algorithm. Once your teammates complete their jobs, you can begin testing with real data. Print the results in a table that is visually appealing. The table should be numbered, and include each quarterback's name, team, overall completion percentage, total yards thrown, and total touchdowns thrown. Below the table, print out the high, low, and average of each category. Be sure that the clumns are labeled and aligned.

Very Important: Your team will need to create a common data format that the person reading the text file, the person running the computations, and the person writing the user interface all agree on. If you are all working in different data formats, then it will be hard for your individual contributions to interface with each other.

Some things to consider: How do you bring all of your code together? Email? Dropbox? Something else? How do you resolve disagreements? There are likely to be bugs in each individual component. There will likely be more bugs when you bring them together? How do you work through and solve these?

Programming Style:

This course does not have a required coding style (at least not yet). Your code should be nicely arranged with proper whitespace, along with appropriate and consistently named variables/functions. Any code that is executed repeatedly should be put into its own function. This makes it easier for you and others to maintain the code.

Homework Submissions:

Please upload your source code to Blackboard. There should only be a single .cs file for this project. You'll find the upload page by clicking "Project #3: Fantasy Football Geeks" on the Assignments page. I will cut-and-paste your code into a scratch project in order to review it, compile it, and test it.

Class Presentations:

You will need to present your code to the class. You must talk through the most challenging part of the project: show this portion of your code, explain the problem you encountered and how you went about solving it. You will only have 2-3 minutes to demonstrate which is NOT MUCH TIME.