CS424 Programming Languages — Programming Assignment #1

Due: See Canvas for Due Date — (40 points)

Problem Statement

You are to develop a program in the Go programming language to read Quarterback data from an input file into a list/collection. You will print them sorted by player name (last name, first name.

Each player will have a line of data in the data file in the following format:

firstname lastname followed by a list of 5 integers.

These five integer values represent: Completions, Attempts, Yards, Touchdowns, Interceptions.

You may assume there are no errors in the data. I will not leave data out or put non-numeric data in numeric fields. You must read until the end of the file is found. There will be no blanks at the end of a line of data and there are no blank lines, or blank lines at the end of the data set.

You will compute a passer rating and completion percentage for each quarterback based on his stats. You will display a report to the screen (see below for exact format of the report.)

Summary of Operation

- Prompt the user for the input file name. DO NOT hardcode file names into your program.
- Open input file
- Read each player in and add them to your data set. Compute their passer rating and completion percentages. [*Formulas given below*]
- Keep track of the number of players in the list.
- Write a report summary line to the screen.
- Include the player with the highest rating in the summary at the top (See sample)
- Write each item from the list formatted to the screen, along with any other output required by the assignment. Format your numeric data as shown in the examples.

TURN IN:

Submit the electronic version of your project on CANVAS. Make sure your name and the name of the file is in the comments at the top of each program file, along with the system you tested your program on. It must work on at least version 1.15 (1.16 is installed on some of our PCs and on the cs linux systems). Please put all of your program solution In a single main program file, with the **.go** filetype. Do not upload the project.

NOTE:

I am not teaching the Go language in class. It is up to you to give yourself enough time to explore and learn the language in order to complete this assignment.

The Go language tools can be downloaded from https://golang.org/

There is a tutorial for learning the basics of Go at: https://tour.golang.org/

Other Requirements

- Your program must be well-commented. Comment all variables, functions and remember to have a section of comments at the top of your program that includes your name, date, course section and a description of what your program does.
- Use good variable names.
- Think about data objects and what they contain. Even though GO is not fully object-oriented (as in inheritance) we should be using good design practices.
- Use good and consistent naming conventions for data members.
- Use proper code indentation to make sure your program is easy to read and understand.
- You may use the following GO import libraries if needed: errors, fmt, bufio, os, strconv, strings, sort. Any others need approval.

Sample Execution (user input shown in yellow)

```
Welcome to the passer rating test program.  I am going to read player
statistics from an input data file.  You will tell me the name of
your input file.

Enter the name of your input file:  playerinput.txt

QUARTERBACK REPORT --- 10 PLAYERS FOUND IN FILE
HIGHEST PASSER RATING - Jackson, Lamar

    PLAYER NAME      :    Rating   Comp%
-----------------------------------------
         Allen, Josh :     101.4    63.6
         Burrow, Joe :     108.5    70.6
         Carr, Derek :     101.0    67.7
     Daniels, Jayden :     100.1    69.0
         Goff, Jared :     111.2    72.4
      Hebert, Justin :     101.7    65.9
        Hurts, Jalen :     103.7    68.7
      Jackson, Lamar :     119.6    66.7
    Mahomes, Patrick :      93.5    67.5
        Purdy, Brock :      96.1    65.9

End of Program 1
```

Sample Input File for the above report

```
Jayden Daniels 331 480 3568 25 9
Justin Hebert 332 504 3870 23 3
Joe Burrow 460 652 4918 43 9
Jared Goff 390 539 4629 41 16
Lamar Jackson 316 474 4172 41 4
Brock Purdy 300 455 3864 20 12
Patrick Mahomes 392 581 3928 26 11
Josh Allen 307 483 3731 28 6
Jalen Hurts 248 361 2903 18 5
Derek Carr 189 279 2145 15 5
```

How to compute Passer Stats:

Completion Percentage = COMP / ATT

Computing Passer Rating is based on 4 values:

$$value1 = \left[\frac{COMP}{ATT} - 0.3\right] \times 5$$

$$value2 = \left[\frac{YDS}{ATT} - 3\right] \times 0.25$$

$$value3 = \frac{TDS}{ATT} \times 20$$

$$value4 = \frac{0.095 - \frac{INT}{ATT}}{0.04}$$

$$passer\ rating = \frac{\sum value_i}{6} \times 100$$

Abbreviations:

- *TDS = Touchdowns*
- *YDS = Yards*
- *COMP = Completions*
- *ATT = Attempts*
- *INT = Interceptions*

References:

I am using player passing statistics found at http://www.nfl.com/stats/player

General Grading Rubric (out of 40 points)

I provide this as a general guideline for how each area of a program is to be rated for grading purposes.

Performance Element	5 - Excellent	4 – Very good	3 - Good	2 - Limited	1 - Inadequate
Specifications ***(20 points)***	Program runs & meets all specifications	Runs, gets correct answers, output displayed correctly, meets most other specifications	Runs, gets correct answers, output is not displayed correctly or other specifications not met	Runs, get some correct results, does not meet most specifications	Program does not run, runs but gets no results or mostly wrong results
Readability ***(5 points)***	Code is well organized and easy to follow	Most of the code is well organized & easy to read	Parts of the code are easy to read but the organization is not good	Code is readable if the reader knows what it is supposed to be doing	Code is poorly organized and very difficult to read
Documentation (i.e. comments) ***(5 points)***	Documentation is clearly written & explains what the program as a whole should do; comment blocks introduce each function	Documentation is brief but helpful, includes header information for functions	Documentation consists of embedded comments and some header information for functions	Little or no documentation other than obvious embedded comments	Little or no documentation
Software Architecture ***(10 points)***	Displays excellent modularity and code structure.	Displays good information modularity, logic is not copy/paste	Displays some structural modularity, but code is not significantly structured into functions	Little or no structure	Displays excellent modularity and code structure.