1 Instructions

You may work in **pairs** (that is, as a **group of two**) with a **partner** on this lab project if you **wish** or you may work **alone**. If you work with a partner, only submit **one** lab project with **both** of your **names** in the **source code file** to Blackboard for grading; you will each earn the **same** number of points. **What** to hand in, and **by when**, is discussed in **Section 5**; read it.

2 Lab Objectives

After completing this assignment the student should be able to:

- Complete all of the objectives of the previous lab projects.
- Write **function definitions**, **call functions**, **pass parameters**, **define local variables**.
- Open a **text file for reading/writing** and read/write strings and numbers from/to the text file.

3 Prelab Exercises

- First, skip to Section 4 and read the lab project **software requirements**. Then come back here.
- Create a new Code::Blocks C++ project named *Lab05*.
- Navigate to the course website and download the *Lab05.cpp* file. This file is a template containing most of the code for the lab project, but in various places, the code has not been completed. Your job shall be to complete the code by reading the comments and writing proper C++ code in the locations indicated by ??? symbols. Follow the instruction in Steps 28 through 32 of the Code::Blocks tutorial[1] to add *Lab05.cpp* to your Code::Blocks project.
- Navigate to the course website and download the *stats-season.txt* file. This program will read input data from this file and will write data to an output file named *stats-game.txt*. Copy *stats-season.txt* to your Code::Blocks *Lab05* project folder. For example, when you created your *Lab05* project, if you specified *C:\cse100\Lab05* as the project folder, then copy *stats-season.txt* to *C:\cse100\Lab05*.
- When you run your program, you will see the output window appear, but nothing of substance will be displayed in it. This is because the output is being sent to the output file named *stats-game.txt* rather than to the output window. To know if your program worked correctly, you need to open the output file and examine its contents. Click **File | Open** on the main menu. In the **Open** dialog, navigate to your project directory *C:\cse100\Lab05* where you should see a file named *stats-game* or possibly *stats-game.txt*. Click on that file name and then click the **Open** button. Verify the output file contains the correct contents.

4 Lab Exercise

A text file named *stats-season.txt* contains season statistics for a basketball team, e.g.,

```
Name         GP  FGM  FGA  FTM  FTA  3M  3A  PTS  REB
F.Flintstone 23  88   190  50   82   27  74  253  100
B.Rubble     23  85   212  45   53   24  84  239  94
W.Flintstone 14  56   101  29   46   0   1   141  82
H.Simpson    23  77   118  62   107  1   3   216  143
N.Flanders   23  57   127  15   17   37  91  166  51
N.Explosion  23  53   109  9    14   46  95  161  63
K.Broflovski 21  31   100  57   79   8   29  127  38
E.Cartman    18  28   66   10   12   17  37  83   22
G.Jetson     17  26   58   19   29   3   16  74   37
S.Skwigelf   18  19   43   18   25   3   9   59   20
T.Wartooth   22  19   59   23   40   7   23  68   46
```

Where the contents of each column is,

Name	The players name, formatted as *I.s* where *I* is the initial of the player's first name and *s* is the player's surname.
GP	Games played.
FGM	Field goals made.
FGA	Field goals attempted.
FTM	Free throws made.
FTA	Free throws attempted.
3M	Three point field goals made.
3A	Thee point field goals attempted.
PTS	Points (the total points scored in the season).
REB	Rebounds.

As team statistician, your job is to write a C++ program which reads *stats-season.txt*, calculates per game statistics, and writes the per game statistics to an output file named *stats-game.txt*. The per game statistics that shall be calculated are,

1 http://devlang.com/cse100_codeblocks

PPG	Points per game. PTS divided by GP.
RPG	Rebounds per game. REB divided by GP.
FG%	Field goal percentage. FGM divided by FGA.
FT%	Free throw percentage. FTM divided by FTA.
3P%	Three point field goal percentage. 3P divided by 3A.

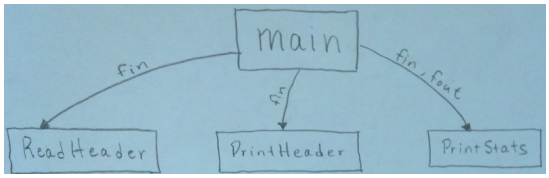
For this particular *stats-season.txt* input file, the contents of *stats-game.txt* will be,

```
Name              GP   PPG   RPG    FG%    FT%    3P%
F.Flintstone      23  11.0   4.3  0.463  0.610  0.365
B.Rubble          23  10.4   4.1  0.401  0.849  0.286
W.Flintstone      14  10.1   5.9  0.554  0.630  0.000
H.Simpson         23   9.4   6.2  0.653  0.579  0.333
N.Flanders        23   7.2   2.2  0.449  0.882  0.407
N.Explosion       23   7.0   2.7  0.486  0.643  0.484
K.Broflovski      21   6.0   1.8  0.310  0.722  0.276
E.Cartman         18   4.6   1.2  0.424  0.833  0.459
G.Jetson          17   4.4   2.2  0.448  0.655  0.188
S.Skwigelf        18   3.3   1.1  0.442  0.720  0.333
T.Wartooth        22   3.1   2.1  0.322  0.575  0.304
```

Note that PPG and RPG are printed with 1 digit after the decimal point and percentages are displayed with 3 digits after the decimal points. The names are printed left-justified in a column and the numerical values are printed right-justified in each column.

4.2 Software Design

Here is the structure chart for the program,



main() first calls *ReadHeader*() passing the file input stream object *fin* as an argument[2]; *ReadHeader*() is a void function which returns nothing. Next, *main*() calls *PrintHeader*(), again, passing *fin* as the argument; *PrintHeader*() is also a void function. Finally, *main*() calls *PrintStats*() passing *fin* and the file output stream object *fout* as the arguments; *PrintStats*() is also a void function. Here is the pseudocode,

```
Program Lab5

    Function main () Returns 0
        Define an ifstream object named fin and open "stats-season.txt" for reading
        Verify that fin was successfully opened; if not, terminate the program
        Define an ofstream object named fout and open "stats-game.txt" for writing
        Configure fout so real numbers will be printed in fixed notation
        Call ReadHeader(fin)
        Call PrintHeader(fout)
        Call PrintStats(fin, fout)
        Close fin
        Close fout
        Return 0
    End Function main

    Function PrintHeader (pFout : ofstream by-ref) Returns Nothing
        Print "Name" left-justified in a field of width 16
        Print "GP" right-justified in a field of width 4
        Print "PPG" right-justified in a field of width 6
        Print "RPG" right-justified in a field of width 6
        Print "FG%" right-justified in a field of width 7
        Print "FT%" right-justified in a field of width 7
        Print "3P%" right-justified in a field of width 7
        Print newline
    End Function PrintHeader
```

2 Stream variables are passed to functions using a C++ parameter passing technique known as pass by-reference. We will discuss pass by-reference later in the course, but if you look at the code you will see the function header for ReadHeader() is `void ReadHeader(ifstream& pFin)`. The & indiccates that *fin* is being passed by-reference. Similary, *fout* is passed to *PrintStats*() by-reference as well. Stream objects such as *cin*, *cout*, *fin*, and *fout* must always be passsed by-ref.

```
Function PrintStats (fin : ifstream by ref, fout: ofstream by-ref) Returns Nothing
    Loop 11 times -- we will discuss what a loop is later in the course. Don't worry about it for now.
        Define string object named name and read player name from fin into name
        Define int variables gp, fgm, fga, fta, threem, threea, pts, reb
        Read from fin into gp, fgm, fga, ftm, fta, threem, threea, pts, reb
        Double fgp ← fgm ÷ fga              -- Hint: typecast fgm to double
        Double ftp ← ftm ÷ fta              -- Hint: typecast ftm to double
        Double threep ← threem ÷ threea     -- Hint: typecast threem to double
        Double ppg ← pts ÷ gp               -- Hint: typecast pts to double
        Double rpg ← reb ÷ gp               -- Hint: typecast reb to double
        Print name left-justified in a field of width 16 to fout
        Print gp right-justified in a field of width 4 with 1 digit after the decimal pt to fout
        Print ppg right-justified in a field of width 6 with 1 digit after the decimal pt to fout
        Print rpg right-justified in a field of width 6 with 1 digit after the decimal pt to fout
        Print fgp right-justified in a field of width 7 with 3 digits after the decimal pt to fout
        Print ftp right-justified in a field of width 7 with 3 digits after the decimal pt to fout
        Print threep right-justified in a field of width 7 with 3 digits after the decimal pt to fout
        Print newline
    End Loop
End Function PrintStats

Function ReadHeader (fin : ifstream by-ref) Returns Nothing
    Define string object named header
    Read a line of text from fin into header
End Function ReadHeader

End Program Lab5
```

4.3 Additional Programming Requirements

1. Update the header comment block in the source code template with your author information, your lab date and time, your lab TA, and the two test cases you are to write for the prelab exercise.
2. Carefully **format** your code and follow the **indentation** of the text as shown in the example programs of the textbook.

5 What to Submit for Grading and by When

Upload the *Lab05.cpp* C++ source code file to Blackboard using the lab submission link by the deadline. If your program does not compile or run correctly, upload what you have completed for grading anyway (you will generally receive some partial credit for effort). The deadline for the complete lab project is **4:00am Sat 3 Oct**. Consult the online syllabus for the late and academic integrity policies.

6 Grading Rubric

1. Lab Exercise Program (0 to 5 pts)

a. If the submitted program does, or does not, compile and the student completed less than 50% of the required code correctly, assign **+2 pts**.
c. If the submitted program does not compile and the student completed more than 50% of the required code correctly, assign **+3 pts.**
d. If the submitted program compiles and the student completed more than 50% of the required code correctly, assign **+4 pts**.
e. If the submitted program compiles and is implemented perfectly, or close to perfect with only one or two minor mistakes, assign **+5 pts**.

2. Deadline was 4:00am Sat 3 Oct

1. Assign 20% bonus calculated on the earned pts for a submission prior to 4:00am Thu 1 Oct.
2. Assign 10% bonus calculated on the earned pts for a submission between 4:00am Thu 1 Oct and 4:00am Fri 2 Oct.
3. Deduct 0.5 pt for a submission between 4:00am Sat 3 Oct and 4:00am Sun 4 Oct.
4. Deduct 1 pt for a submission after 4:00am Sun 4 Oct.