CS 215 – Fall 2017

Project 3

Version 1.2 [Max #Games is 15, not 40]

Learning Objectives:

- Developing a small application using Object-Oriented Design, including use of Classes and Objects.
- Use of Partial Arrays to store data.
- Searching and Sorting of arrays.
- Reading data from and writing data to a text file.

General Description:

Write an application that allows the user to enter, view, search and sort statistics on a team of basketball players. The program should start by reading data from a file called **"bballin.txt",** and should end by writing data to a file called **"bballout.txt".**

A **Statistic** (stat) in the program consists of two numbers:

- *number of points made*
- *number of points attempted.*

A percentage is calculated with the formula (100*made)/attempted, rounded off. When the points attempted is zero, the percentage is zero.

Percentages are printed in the format *mm***/***aa* **(***nnn***%)** where *mm* is the points made, *aa* is the points attempted, and *nnn* is the percentage [note: always length 12]. Examples:

```
 0/ 0 (  0%)
12/12 (100%)
 5/10 ( 50%)
```

A **Player** has the following data kept:

- *name*
- *jersey number*
- *number of games played* (max **15**) – [note: this number will be the same for all players]

and three *lists of stats*, one stat per game per list:

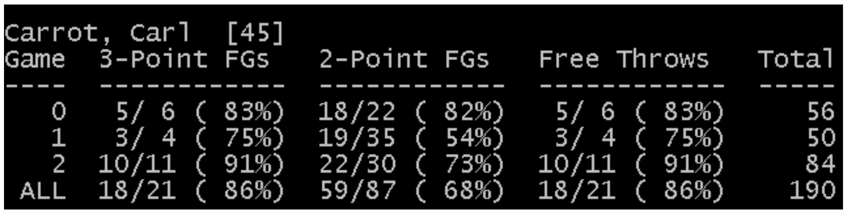
- *3-point field goals*
- *2-point field goals*
- *free throws*

Other data can be *calculated* from these stats:

- A stat of the overall 3-pointers (sum of 3-pointers made/attempted in all games).
- A stat of the overall 2-pointers (sum of 2-pointers made/attempted in all games).
- A stat of the overall free throws (sum of free throws made/attempted in all games).
- A list of total points scored, one for each game, calculated as:

  3-pointers-made * 3 + 2-pointers-made * 2 + free-throws-made

Printing a report for a player should look like this example. The name is "as is" read from the file. The jersey number is in square brackets.



```
Carrot, Carl  [45]
Game  3-Point FGs   2-Point FGs   Free Throws   Total
----  ------------  ------------  ------------  -----
   0   5/ 6 ( 83%)  18/22 ( 82%)   5/ 6 ( 83%)     56
   1   3/ 4 ( 75%)  19/35 ( 54%)   3/ 4 ( 75%)     50
   2  10/11 ( 91%)  22/30 ( 73%)  10/11 ( 91%)     84
 ALL  18/21 ( 86%)  59/87 ( 68%)  18/21 ( 86%)    190
```

A **Team** consists of:

- A list of Players
- The number of players (max 10)
- The number of games (max 15)

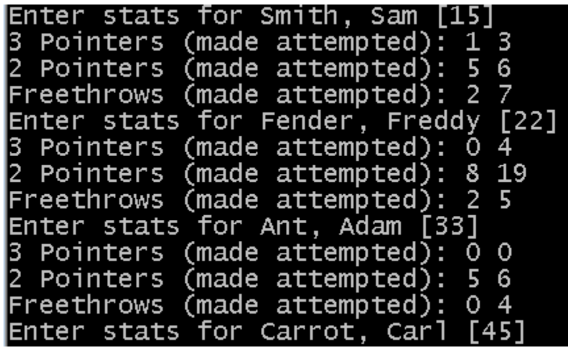
A user should be able to perform the following operations (one method each) on a team:

Print the Team Report:
Print the stat report for all players with a blank line between each player. Use the report format shown above. The program should *pause* after printing the last player.

Add a game to each player on the Team:
The user is asked to enter the stats for a new game for every player in the list.
For each player, it should:

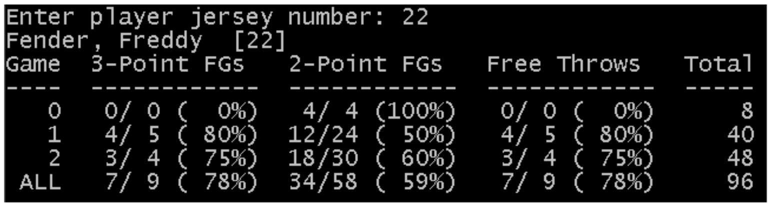
- Display the player's name and jersey number in square brackets.
- Ask the user to enter the shots made and attempted for
    - 3-pointers
    - 2-pointers
    - Free throws
    -

If the program already has the maximum number of games allowed, it should print a message and *not* add a new game.

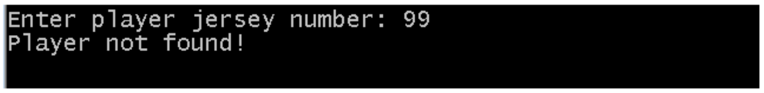


```
Enter stats for Smith, Sam [15]
3 Pointers (made attempted): 1 3
2 Pointers (made attempted): 5 6
Freethrows (made attempted): 2 7
Enter stats for Fender, Freddy [22]
3 Pointers (made attempted): 0 4
2 Pointers (made attempted): 8 19
Freethrows (made attempted): 2 5
Enter stats for Ant, Adam [33]
3 Pointers (made attempted): 0 0
2 Pointers (made attempted): 5 6
Freethrows (made attempted): 0 4
Enter stats for Carrot, Carl [45]
```

Display a player by jersey number:
The program should ask the player to enter the jersey number of the player to find.
When a player with the entered number is not found, it should print "Player not found".
When the player is found, it should print the player's stat report as usual.



```
Enter player jersey number: 22
Fender, Freddy  [22]
Game  3-Point FGs   2-Point FGs    Free Throws   Total
----  ------------  ------------   ------------  -----
   0   0/ 0 (  0%)    4/ 4 (100%)   0/ 0 (  0%)      8
   1   4/ 5 ( 80%)  12/24 ( 50%)    4/ 5 ( 80%)     40
   2   3/ 4 ( 75%)  18/30 ( 60%)    3/ 4 ( 75%)     48
 ALL   7/ 9 ( 78%)  34/58 ( 59%)    7/ 9 ( 78%)     96
```



```
Enter player jersey number: 99
Player not found!
```

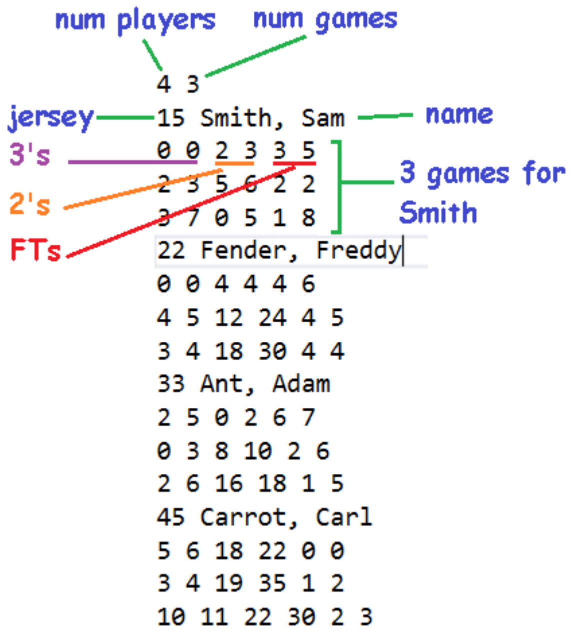
Remove a player from the Team:
The program should ask the player to enter the jersey number of the player to remove. When a player with the number entered is found, the program removes the player from the list and prints a message "Player *name* [*jersey*] removed". When the player is not found, the program prints "Player not found".

Sort Team by Name:
The program sorts its internal list of players by name (names "as is"…no need to "parse" the name for first/last/etc.). It simply prints a message "Players sorted by name" on the screen.

Sort Team by Jersey:
The program sorts its internal list of players by jersey number. It simply prints a message "Players sorted by jersey" on the screen.

Read Team data:
On startup, the program should read the current team data from a file named **“bballin.txt”**. The format of the file is as follows:

- The first line contains the *number of players* and *number of games.*
- For each player:
    - The first line contains the *jersey number* and *name*. The *name* may contain spaces.
    - For each game, there is one line of six numbers:
        - 3-pointers made
        - 3-pointers attempted
        - 2-pointers made
        - 2-pointers attempted
        - Free throws made
        - Free throws attempted

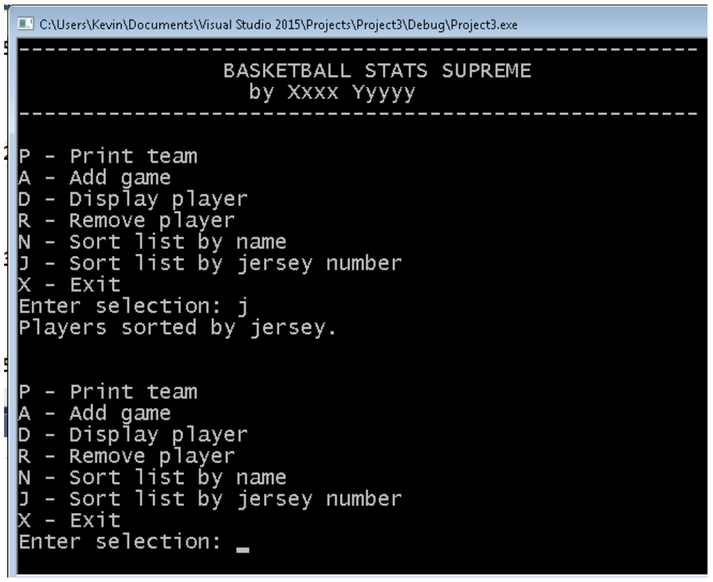


```
4 3
15 Smith, Sam
0 0 2 3 3 5
2 3 5 6 2 2
3 7 0 5 1 8
22 Fender, Freddy
0 0 4 4 4 6
4 5 12 24 4 5
3 4 18 30 4 4
33 Ant, Adam
2 5 0 2 6 7
0 3 8 10 2 6
2 6 16 18 1 5
45 Carrot, Carl
5 6 18 22 0 0
3 4 19 35 1 2
10 11 22 30 2 3
```

Write Team Data:
When the program ends, it should write the current data to a text file named **“bballout.txt”** in the exact same format at the input file.

Overall Operation:
The program should start by printing a logo with your name and section.

It should then display a menu of options and allow the user to an option. The program performs the selected option. When an invalid option is chosen, the program prints an error message. This repeats until the user enters the **exit** option.



```
C:\Users\Kevin\Documents\Visual Studio 2015\Projects\Project3\Debug\Project3.exe
-----------------------------------------------------
               BASKETBALL STATS SUPREME
                  by Xxxx Yyyyy
-----------------------------------------------------

P - Print team
A - Add game
D - Display player
R - Remove player
N - Sort list by name
J - Sort list by jersey number
X - Exit
Enter selection: j
Players sorted by jersey.


P - Print team
A - Add game
D - Display player
R - Remove player
N - Sort list by name
J - Sort list by jersey number
X - Exit
Enter selection: _
```

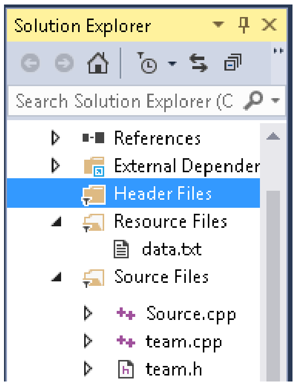
Specifications:
The main() function (along with a couple of others) are provided on the course website. You must use the code provided **as is**, without making any changes! (You can always email the instructor to ask permission to make changes, as usual).
Exceptions:

- Change the logo as you wish. Make sure it contains your name and section.
- Change the program comment box at the top as needed.
- Add any more #includes that might be needed.
-

You should not add any more code to this file. This is your main.cpp (or source.cpp).

You will note that the main() uses a class called **team**, and that it invokes methods inside an object of this class. Therefore, you should:

- Add **team.h** header file to the project. Code the class interface there.
- Add **team.cpp** source file to the project. Code the class implementation there.
- Use the exact spelling of the class and methods as used in the main()



You will of course add other header and source files to this project. Give them good names!

Some general rules about writing classes:

- The header and source files should have the same name as the class.
```
class team {
          …
};
```
  Use **team**.h and **team**.cpp
- There should be a comment box at the top of each header file with the name of the class and a brief description:
```
//-------------------------------------
//                  class team
//-------------------------------------
// Implements a team of …
//-------------------------------------
```
- There should be at least a comment box at the beginning of each method .
  Exceptions:
  - Very small methods may be listed under one comment box. Example: a list of short set() methods may have one box with **set methods** centered in the comment box.
- Methods should be no longer than 30 lines between the beginning and ending braces { … }
- All data members should be initialized to a default “empty” value. (ie. 0, “”)

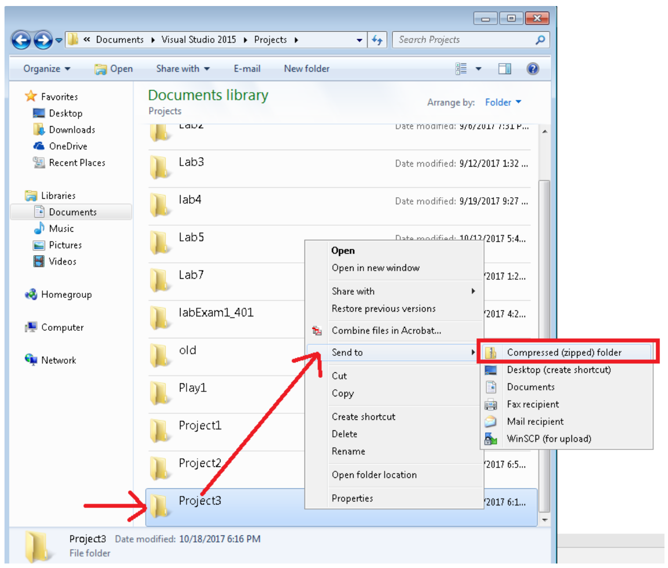
- There should be a **set()** method and a **get()** method for each member.
    - Each set() should validate the given arguments when needed (ex: points made <= points attempted). When arguments are invalid, do not change the members and print an error message.
    - set()s may be combined where it makes sense. Ex: for a stat, set(made,attempted) is good.
    - Exception: if you need an “internal use only” member for a class, set() and get() are not required.

Specific Requirements for this project:

- Do not use file streams (ifstream, ofstream) in any method besides the read() and write() methods of class **team.** Use methods of any sub-objects to set() and get() data from them.

Submission:

Since this project contains multiple source files, it will be easier to **zip (compress)** the entire project folder into a **.zip** file and submit that. Be sure to zip the *entire* project folder as shown below.



Canvas will only accept .zip files for submission for this project.

- Using Windows Explorer, find the folder (probably under **Visual Studio 2015\Projects**) where all your project and lab folders are.
- **Right Click** on your **Project 3** folder. A dialog box should pop up (it may be blank for a few moments…patience!)
- **Click** on **Send To**. Another dialog should appear.
- **Click** on **Compressed (zipped) folder**.
- After a few moments, a new .zip file will appear in that directory.
- **Rename** the zip file to *yourlastname_yourfirstname*.zip
- Submit that .zip file in Canvas.