CSCI 1300 CS1: Starting Computing
Ashraf, Cox, Spring 2020
Homework 7
Due: Saturday, March 14, by 6 pm
(5 % bonus on the total score if all three parts are submitted by 11:59 pm March 13)

Objectives

- Understand classes and objects

You can find **hw7 note: classes and objects** on Moodle.

Submissions

- **Conceptual reviews(mcq)**. There are a few multiple-choice questions to check your conceptual understanding. Don't forget to complete them!
- **h files and C++ files**. All files should be named as specified in each question, and they should compile and run on Cloud 9 to earn full points. TAs will be grading styles of your code and comments. Please see the style guide on Moodle and the summary note on Moodle. At the top of each file, write your name with the following format:

  ```
  // CS1300 Spring 2020
  // Author: Punith Sandhu
  // Recitation: 123 – Favorite TA
  // Homework 7 - Problem # …
  ```

  For each question, you should have a h file and cpp file nicely organized. In a driver file, you need to create a program that calls the class methods in the main to test each method you write. **Here is an example**.

- **Code runner**. Your program will be graded by the code runner. You can modify your code and re-submit (press Check again as many times as you need to, up until the assignment due date.

Questions

Question 1(15pt): Player Class
Create `Player.h` and `Player.cpp` to implement the `Player` class as described below. You will also need to create a `playerDriver.cpp` file to test your Player class implementation.

The `Player` class has the following attributes:

Data members (private):	
`name`: `string`	The player's name
`points`: `double`	The player's points
Member functions (public):	
Default constructor	Set `name` to an empty `string` and `points` to value 0.
Parameterized constructor	Takes a `string` and `double` assigning `name` and `points`, in this order
`getName()`	Returns the player's `name` as a `string`
`getPoints()`	Returns the player's `points` as a `double`
`setName(string)`	Sets the player's `name` (and returns nothing)
`setPoints(double)`	Sets the player's `points` (and returns nothing)

For the zip submissions, the files should be named as `Player.h`, `Player.cpp`, and `playerDriver.cpp`. Your implementation should be organized nicely into separate files. In your `playerDriver.cpp`, you should have a main `main` where you write test cases for each method (constructor, getters, and setters). Please check the sample submissions on Moodle

Sample main (Be sure to test all methods!)	Expected outputs

`Player hector(“Hector”, 6.2);`
`cout << hector.getName() << endl;`
`cout << hector.getPoints() << endl;`
	Hector
	6.2

For the code runner, paste your Player class and its implementation (both `Player.h` and `Player.cpp`)

Question 2(25pt): Team Class
Create `Team.h` and `Team.cpp` to implement the `Team` class as described below. You will also need to create a `teamDriver.cpp` file to test your `Team` class implementation.

The `Team` class has the following attributes:

Data members (private):	
`teamName`: `string`	The team's name
`players`: an array of `Player` objects of size 50	The players on this team
`numPlayers:int`	The number of players stored in the array
Member functions (public):	
Default constructor	Set `name` to an empty `string` and numPlayers to 0
Parameterized constructor	Takes a string to initialize `teamName` and sets numPlayers to 0.
`setTeamName(string)`	Takes a `string` to set `teamName` (and returns nothing)
`readRoster(string)`	Takes the file name (a `string`), reads a list of player names and their points values, and stores them into the `players` array for this team. It returns the number of players in the file as an `integer`.
`getPlayerName(int)`	Returns the `name` (`string`) of the player at position `i` within the `players` array
`getPlayerPoints(int)`	Returns the `points` (`double`) of the player at position `i` within the `players` array
`getNumPlayers()`	Returns the number of players on this team (as an `integer`)
`getTeamName()`	Returns the name of this team (`string`)

Method specifications:
`Team` Class Method: **`readRoster(string)`**: It takes a file name and reads a list of players and their points separated by a comma. Each player should be stored into the `players` array. The function returns the number of players stored in the array. If the file cannot be opened, it should return -1.

Sample file (**roster1.txt**):

```
O'Flaherty,5.5
Ioana Fleming,6.1
Patil,8
Ku,4.9
Sankaralingam,1.7
```

`Team` Class Method: **`getPlayerName(int)`** / **`getPlayerPoints(int)`**: They take the index of a player object within the `Player` array.

For **`getPlayerName(int)`**, it returns the name of the player at the given index. If the index is invalid, then it returns “ERROR”.

For **`getPlayerPoints(int)`**, it returns the point of the player at the given index. If the index is invalid, then it returns -1.

What’s an invalid index? An invalid index means that there is no player at that index. There are two cases in which this can occur: 1) The index is greater than or equal to the number of players in the team. 2) the index is not within the bounds of the array.

For the zip submissions, the files should be named as `Team.h`, `Team.cpp`, and `teamDriver.cpp`. Your implementation should be organized nicely into separate files. In the `main`, you should have test cases to test all of your functions.

Sample main (Be sure to test all methods!)	Expected outputs

`// Using roster1.txt from Moodle`
`Team team1(“Seg Faultline”);`
`cout << team1.getTeamName() << endl;`
`team1.readRoster(“roster1.txt”);`
`int n1 = team1.getNumPlayers();`
`cout << n1 << endl;`
`for (int i = 0; i < n1; i++) {`
`    cout << team1.getPlayerName(i) << “ ”;`
`    cout << team1.getPlayerPoints(i) << endl;`
`}`
	Seg Faultline
	5
	O'Flaherty 5.5
	Ioana Fleming 6.1
	Patil 8
	Ku 4.9
	Sankaralingam 1.7

For the code runner, paste your `Player` class and its implementation and `Team` class and its implementation (both `Team.h` and `Team.cpp`)

Question 3(20pt): the game() function
Write a function, game, that takes two Team objects as parameters and returns the name of the winning team.

Function specifications:

- The function name: **game**
- The function takes two parameters, both of the type `Team`.
- The function returns the name of the winning team, as a `string`
    - To determine the winning team, add up the points associated with each team's first 4 players. The team with more points is the winner.
    - If one or both of the teams do not have 4 or more players, your function should return "forfeit"
    - If the teams have the same total points, your function should return "draw"
    - The output of "forfeit" takes higher priority than "draw" in the sense that if a team doesn't have enough players, a game cannot be played, so there could be no draw (for example, if both teams have no players, this would constitute a "forfeit", not a draw).

Example1, the team Seg Faultline wins because the total points of their first 4 players is 24.5 points, whereas Team Maim only has 21.9 points.

Sample main (Be sure to test all methods!)	Expected outputs

`// Using roster1.txt and roster2.txt`
`// from Moodle`
`Team team1("Seg Faultline");`
`team1.readRoster("roster1.txt");`
`Team team2("Team Maim");`
`team2.readRoster("roster2.txt");`
`string winner = game(team1, team2);`
`cout << "The winner is: " << winner << endl;`
	The winner is: Seg Faultline

The file name should be named `gameDriver.cpp`. In your main, make sure that you call the function and output the return value.

For Coderunner, paste your `Team` and `Player` classes and their implementations, and your `game` function (submit what you did for Problems 2 and 3, and add your `game` function).

Homework 7 checklist

Here is a checklist for submitting the assignment:

1. Complete the **Conceptual reviews(mcq)**
2. Complete the code **Homework 7 CodeRunner**
3. Submit one zip file to **Homework 7 zip file submission**. The zip file should be named, **hmwk7_lastname.zip**. It should have the following 7 files:
   - **Player.h**
   - **Player.cpp**
   - **playerDriver.cpp**
   - **Team.h**
   - **Team.cpp**
   - **teamDriver.cpp**
   - **gameDriver.cpp**

Homework 7 points summary

Criteria	**Pts**
Conceptual reviews (MCQ)	10
CodeRunner (problem 1 - 3)	60
C++ file submission (compiles and runs, style and comments)	30
Recitation attendance (March 9, March 10)*	-30
Total	100

* if your attendance is not recorded, you will lose points. Make sure your attendance is recorded on Moodle.