CS 211 Spring 2021 – Data Structures
Lab for Week #03

For this lab exercise, you will work in 2-person teams (a single 3-person team will be allowed is there are an odd total number of students in the lab). The point here is to have teams discuss the lab and offer each other support in making sure the IDE of your choice is working!

For this lab, **you will take turns** in being the "driver" (the team member sharing the screen with the IDE) and the "navigator" (the team member who is viewing the editing being done and collaborating in performing the Lab Exercises). **Each student will use the IDE of their choice** to compile and test the C++ code to ensure the IDE is up to performing the functions necessary to complete labs and assignments.

Lab Exercise

This lab builds on the work done in the Week 02 Lab on the PlayerChar class.

- In the Canvas lab page, you'll find links to the files **`PlayerChar.cpp`**, **`PlayerChar.h`**, **`PlayerTeam.cpp`**, **`PlayerTeam.h`** and **`main.cpp`** – download those files, and copy the contents of those files into the appropriate places in your IDE for C++ and H files. They contain new and refactored code, so use this code instead of what you wrote in the previous lab!

- At the top of **each** file, document your module as follows based on your 2-person team:

  ```
  # CS 211 Spring 2021 – Week 03 Lab
  # <Student_Name1> and <Student_Name2>
  ```

  Substitute your own names for the **`Student_Name`** placeholders above.

- Define a method called **restoreHealth** in the **PlayerChar** class that restores the HP (health points) of a **PlayerChar** to its initial (maximum) value – the product of multiplying **strength** and **agility**. Place this method within **PlayerChar.cpp** in the specified place, and use (uncomment) the tests in **main.cpp** to verify its correct execution.

- We're going to set up teams of **PlayerChar** objects that can eventually battle each other (one at a time) until only one team's players remain standing! To get there, we'll first need to define what a **PlayerTeam** is, and that's today's lab exercise.

- Define a new class called **PlayerTeam** whose objects are teams (collections) of **PlayerChar** objects. You'll use C++ arrays for the purpose of holding the **PlayerChar** objects, but that fact will be hidden from the users of the class – you'll be writing methods for interfacing with the PlayerTeam objects.

- Note that the PlayerTeam's private data member definitions includes a POINTER to a **PlayerChar**. This is actually a pointer to an array of PlayerChar objects that you will dynamically allocate in the constructors!

- The **PlayerTeam** class will have two constructors, one that takes a team name of type **string** as its only argument (see the next part for information on team names), and another that takes no argument at all that will set a default **DEFAULT_TEAM_NAME** (a constant that is pre-defined in **PlayerTeam.h**).
  In each constructor, be sure to do the following:
  - Store the team name into the **string** variable that is kept **private**.
  - Dynamically allocate the array of **PlayerChar** objects to handle a team of the appropriate size. The items in the array will be "default" **PlayerChar** objects when first created. Then, by using a method called **setPlayer** each **PlayerChar** in the array can be customized.

- Define a destructor named **~PlayerTeam** (the default name for a destructor) that completely cleans up after itself – it must deallocate the array that stores the **PlayerChar** objects!

- Define methods named **getTeamName** and **setTeamName** as an accessor and a mutator for the team's name.

- Define an overloaded **void** method named **setPlayer** that will customize the **PlayerChar** at a particular position in the **PlayerTeam** array at the supplied position within the team, a "position number" like on a baseball player's jersey, from **0** to **TEAM_SIZE – 1.**

  Here are examples of calls to this method, which will have three overloaded versions, all of which will set a **PlayerChar**'s characteristics at a "position number" within a **PlayerTeam** named **my_team**. These examples place Fred the Ogre (default from **PlayerChar**) as Player #0, Ariel the Sprite (with default strength and agility from **PlayerChar**) as Player #1, and Barney the Dinosaur as Player #2:

  ```
  my_team->setPlayer(0); // Uses all default values of PlayerChar
  my_team->setPlayer("Ariel", "Sprite", 1);
  my_team->setPlayer("Barney", 350, 7.5, "Dinosaur", 2);
  ```

- A **void** method named **resetPlayer** that resets a **PlayerChar** in a **PlayerTeam** to the default. This will reset the player at that position to the default player (Fred the Ogre, as defined in the zero-argument constructor in **PlayerChar**).
  Example: `my_team->resetPlayer(2); // Resets Barney in Pos 2 to default`

- A **void** method named **printTeamPlayer** that expects a position number, and uses the **printPlayer** method in **PlayerChar** to print the info of the player at that position to the screen.

- Use the code in the **main()** function that makes use of these methods and tests their proper function.

  YOU NEED TO SUBMIT **PlayerChar.cpp, main.cpp, PlayerChar.h, PlayerTeam.cpp** and **PlayerTeam.h** for this lab. No other files are needed! It will be tested against a standard set of tests the instructor has to determine its functionality.

ONCE FINISHED, ALL STUDENTS SHOULD SUBMIT ALL THE FILES VIA CANVAS.
Submit ALL the CPP files and H files using the "multiple file submission" feature of Canvas.