CS 112 Fall 2020 – Data Structures
Lab for Week #09

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).

NOTE: If you are using an IDE other than NetBeans, you should ensure that the code you write and test will run in the NetBeans IDE. This rule will apply more strongly for individual homework assignments than for the weekly lab sessions, but understanding what adaptations you need to make to your code to run in NetBeans is a necessary component of your participation in this course. If incompatible code continues to be a problem, the instructor reserves the right to require the use of NetBeans for all assignments.

Lab Exercise

Last week, teams defined a C++ class based on the **PlayerChar** class demonstrated in class in Week 5, Lecture 2. A solution to last week's lab is posted for you to use, as you'll need this code for this week's lab.

Download the provided **main.cpp**, **PlayerChar.cpp**, and **PlayerChar.h** files, along with files named **PlayerTeam.cpp** and **PlayerTeam.h**. These files are partially completed, and you will supply methods in **PlayerTeam.cpp** (and their headers in **PlayerTeam.h**) as specified below.

You will NOT modify the **PlayerChar** files! **ONLY** the **PlayerTeam.cpp** and **PlayerTeam.h** files will be edited to create a new class named **PlayerTeam**. You'll also edit the **main.cpp** file to uncomment lines of code to test the methods you'll write for the PlayerTeam class.

Perform the following tasks:

- At the top of the **PlayerTeam.cpp** and **PlayerTeam.h** files, document your module as follows based on your 2-person team:

  ```
  # CS 112 Fall 2020 – Week 09 Lab
  # <Student_Name1> and <Student_Name2>
  ```

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

- Read the **PlayerTeam.h** and **PlayerTeam.cpp** files closely to see the provided data members and methods. Use the supplied data mambers and identifier names in the code of your methods.

- Define the class named **PlayerTeam** whose objects are a team (an array) 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.

- The **PlayerTeam** class will have three constructor methods that you will write:
  - The first constructor takes two arguments – a **string** for the **PlayerTeam**'s name, and an **int** value for how many **PlayerChar** characters will be on the team. It will **DYNAMICALLY ALLOCATE** the array of **PlayerChar** objects that stores the team members. The array will be "empty" when created, and can use the default copy constructor for the **PlayerChar** class to be filled with **PlayerChar** data.
  - The second constructor takes no arguments. It will name the team using the constant **DEFAULT_TEAM_NAME** and the size will be set to the constant **DEFAULT_TEAM_SIZE**. It will **DYNAMICALLY ALLOCATE** the array of **PlayerChar** objects that stores the team members. The array will be "empty" when created, and can use the default copy constructor for the **PlayerChar** class to be filled with **PlayerChar** data.
  - The third constructor is a COPY CONSTRUCTOR that expects an existing PlayerTeam object. It will **DYNAMICALLY ALLOCATE** the array of **PlayerChar** objects that stores the team members. It will them copy all the data in the passed PlayerTeam object into the new object.
- The **PlayerTeam** class will have one destructor method that you will write:
  - The destructor will **DEALLOCATE** the array of **PlayerChar** objects that stores the team members.
- The **PlayerTeam** class will have one accessor method that you will write:
  - The method **getPlayer** will expect an **int** for the index in the array, and will return the **PlayerChar** stored in the **teamArray** at the given index.
- The **PlayerTeam** class will have one operator method that you will write:
  - The method will overload the comparison operator **==** and will return **true** if the two **PlayerTeam** objects being compared have the same number of **PlayerChar** objects in the **PlayerTeam**, and the same **role** in each corresponding **PlayerChar** in the two **PlayerTeam** arrays. For example, if both **PlayerTeam**s have two **PlayerChar**s, an **Ogre** at index **0** and an **Elf** at index **1**, then the operator will evaluate to true. It will NOT compare other characteristics of the **PlayerChar** objects – only the **role** of each **PlayerChar**.

Use the supplied code in the **main.cpp** file (or similar file) that calls all these methods to test them. Uncomment the lines of code that can test the methods that you write.

It is NOT necessary to submit the **main.cpp** file – the instructor has a **main.cpp** with which to test your code. At the top of the **CPP** and **H** files, be sure to place comments with your names, as usual:

```
// CS 112 Fall 2020 – Week 08 Lab
// <Student_Name1> and <Student_Name2>
```

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

When finished, submit the Week 08 Lab files to the Week 08 Lab assignment page. Use the Multiple File Submission tool in Canvas to submit the **CPP** and **H** files (except the main.cpp file, which does not need to be turned in).