

## CS 150 Lab10 -- Week 13 Week of April /4/2016

You will work alone on the **assignment in lab this week**. You will lose points for work that is not submitted by the end of the lab period. (minus 10 points for a late program) Remember to **include the necessary documentation in the source code and in the output**. For each program, hand in a copy of the source code and copies of the outputs. Your lab TA will tell you how to hand in your work. There are 6 parts – approximately 17 points each.

In lab 8, we worked with an array of structs, and completed functions to work with those arrays.

This week we will work with an **array of classes**, and complete similar functions.

See the lab 8 assignment for the example output required. It should be the same output format.

The objects are structures of type `avatar`, which is a user defined data type. Similar to the players 'alias' in our projects this semester, an avatar is an icon or figure representing a particular person in a computer game. You can see the class definition at the top of the lab template: `Lab_10_spr2016.cpp`. Execute the code in file `Lab_10_spr2016.cpp`; notice that it declares an array named `list_one`, initializes it with random numbers, and prints out the contents of the array. Modify the code by adding the following:

**Part a.** Declare a constant `SIZE_TWO` and initialize it with a value of 20. Declare an array named `list_two` with `SIZE_TWO` components of type `avatar`. Call `fill_it` to initialize `list_two`. Call `printit` to print out `list_two`.

**Part b.** Write a void function called **`reversePrinter`** that prints out the contents of an array in reverse order. The parameters to the function should be an array and the number of elements to be printed. Call the function to print out the names of the avatars in array `list_one` and `list_two` in reverse order. This function should only print out the names.

**Part c.** Write a void function called `tally`. The parameters are a list, the list size, and a target; the function prints out the number of entries in the list at or above the target age. Call the function three times from the main function, prompting the user each time.

**Part d** Write a void function **`get_PlayerInfo`**. The parameters will be the array, and the size of the array, and an input file stream. The function will fill each array entry with the rows in the input file: "playerInfo.txt".

**Part e.** Write a void function called **`print_PlayerInfo`**. The parameters will be the array, and the size of the array. The function will print out all of the player data loaded into the array. See the example code below for the program output. Call the functions **`get_PlayerInfo`**, and **`print_PlayerInfo`**, using `list_two` as the array parameter.

**Part f.** Write a function called `lowVal`. The parameters are a list and the list size; the function returns the value of the player with the lowest lifeForce value entry in the list. Call the function twice from the main program.

```
cout << "the player with the lowest life force value in list_one is "  
<<lowVal ( list_one, SIZE_ONE)<<endl<< endl;
```

```
cout << "the player with the lowest life force value in list_one is "  
<<lowVal ( list_two, SIZE_two)<<endl<< endl;
```