Static Arrays

Complete this worksheet by answering the questions using the given program. Make sure to color or **bold** your answers so they are distinguishable from the question.

Getting Started

- ☐ Make a copy of this document by selecting “Make a Copy” from the File menu.
- ☐ Go to https://replit.com/~ and log in with Gmail using your Weber State email address.
- ☐ Open the code linked below
- ☐ Click **fork**, located in the upper right-hand corner, to create your own copy
- ☐ In your copy, go to the upper right-hand corner, and click **invite**. Then, choose **Generate a Join Link**.
- ☐ Paste the generated link in the second bullet point below

Worksheet Program Links

- *Starter Code*: Repl File Arrays
- *Finished Program Code*: *https://replit.com/join/hmloaltdyb-trinanixon1*

Part 1: Static Arrays

In this section, we will observe and practice the use of static arrays. Navigate to Part 1 in the main() method and answer the following questions.

1. What is the variable n an instance of? Numbers
2. How many times is the DisplayList() function called? 3 times
3. Navigate to the Numbers class. Observe the private data. What is the size of the array?

   10
4. Navigate to the Numbers.cpp file and observe the PopulateList() function. What type of loop is used to iterate through the array, by index, or with a ranged for loop? Why do you think this type of loop is used? An index-based for loop is used which is a good choice because we can access and modify each element in the numbers array. A ranged for loop is used when it’s not necessary to access each index in the array specifically, but instead, just iterate over the elements

5. What is each value of the loop being set to? A random number between 0 and 99.
6. Observe the DisplayList() function What type of loop is used to iterate through the array? By index, or with a ranged for loop? A ranged for loop is used.
7. Navigate to the AddFive() function. Write a loop to add 5 to each of the values in the array. Run the program and verify that the second array of numbers has five more than the first.
8. Navigate to the DisplayBackwards() function. Write a loop similar to the DisplayList() function that displays the values in reverse order. Replace the final call to DisplayList() under part 1 with DisplayBackwards() then run the program and verify that the list is displaying backward.

Part 2: Arrays of Objects

This class is intended to display the list of the top 10 high scores of a game in order from the highest to lowest score. The code uses an array of objects to associate the name of the player with the score. Answer the following questions about the HighScores class and array..

1. Navigate to the HighScores.h file. Observe the player definition. Is it created as a class or struct? Why do you suppose there is not a public and private section of the definition? It's created as a struct. In a struct, members are public by default so there's no need to specify public or private.
2. What two data members does the Player definition contain? String name_ and int score_;
3. What do you think the operator < and operator > allow the program to compare? They allow us to compare two player objects and determine whether one score is less than another
4. Navigate to the HighScores class. What data type does the highScores array contain? int

5. How many scores will the array contain? 10
6. Navigate to the AddScore() function in the HighScores.cpp file. What position of the array is the parameter Player being placed in? It's being placed in the array at the position 'currentNumberScores'. When there are less than 10 scores in the array, the new score is placed at the index 'currentNumberScores', which is the next available position in the array.
7. What happens to that variable after the player's score is added to the array? It gets incremented.
8. When is the score added to the array if currentNumberScores is 10? In the else if statement, if p is greater than the score in the last position of the array, then it needs to be included in the top 10 scores so it needs to be swapped and PushLastScoreUp() is called.
9. Notice the function call to PushLastScoreUp(). The PushLastScoreUp() function will move the score to the correct position in the array based on the score of the player. Take a moment and Observe the code for the PushLastScoreUp() function. What happens to the last score if it is larger than the score above it in the array? The two scores are swapped. It will be in the correct place in the array when it's not larger than the score above it - keeping it in descending order.
10. What happens if it is not larger than the score above it? It moves on to the next 'else' statement which only has a 'return' which exits the loop and the function.
11. Observe the operator<<. Modify the code in the operator to display the array of scores. Display the name, a tab, then the score. You can either add an operator<< to the Player struct for the display, or display the values individually in the loop. We will test our class in part 3.

Part 3: Code the High Scores

It is time to add in a few high scores to test our class.

1. Navigate to the main.cpp file. Observe the function call to AddScore(). Why do you think there are brackets { } surrounding the name and number? The brackets create an instance of the “Player” struct. They give a value for the name_ and score_ members. In C++, brackets are used to initialize the members of the struct.
2. What was the data type for the AddScore() parameter? Players is the data type, so the AddScore() function accepts an instance of the Player struct as the parameter.
3. Create an array of strings that contains at least 10 names.
4. Create an array of int values that contain the same amount of scores for each of the names.
5. Write a for loop to iterate through the names. Move the function call to AddScore() replacing the name and score with your names, and scores array values.
6. Run the program and make sure the names and scores are displayed neatly on the screen.