CS&E 1222

Lab 9 – Arrays Lab Assignment – 20 points

- ✓ The *lab* must be accomplished solely by you:
  - ➢ DO NOT look at anyone's code other than your own, including code from another's student in your section or another section of the course, or any third party source, e.g. the Internet
  - ➢ DO NOT share or copy anyone else's code for any graded assignment
  - ➢ DO NOT work in pairs or groups
- ✓ All cases of academic misconduct will be reported to the *Committee On Academic Misconduct* (COAM).

Setting up the Programming Environment

Effective commenting and tabbing will affect your grade. The "style" of your program should follow the style of the sample programs in the lecture notes. Also see the example code from Lab #1. Your program should have the file name, your name, creation and last modification dates and a brief description of the program in the comments. ***In addition, read the document on "Commenting" found in the Content tab on Carmen under "Tutorials"***.

1. At the Linux command line type `mkdir lab9`. This will create a new directory named **lab9**. Work out of this directory. In order to do that, type `cd lab`. This changes the current working directory to the directory **lab9**.
2. If you have created the directory **lab9**, then just type `cd lab9`.
3. Copy the file **array_solution.exe** from the directory **/class/cse1222/9643/lab9** into the current directory.

```
cp /class/cse1222/9643/lab9/array_solution.exe .
```

Be sure to include ***9643*** (this is your course section indicator) and the period, ".".

Programming Assignment

Write a program called **array.cpp** which prompts and reads a list of non-negative numbers (ints), i.e. can be 0, into an array, prints the array, finds the maximum value in the array, adds the maximum value to each array number and then prints the maximum number and the modified array. The input list is terminated by a -99. The program should have a function read_list() for reading in a list of values, a function print_array which prints each array element, a function find_max() for finding maximum number in the array, and a function array_add() which adds a number from each element of the array.

Run **array_solution.exe** to see an example of the program.

1. All functions should be written AFTER the main procedure.
2. A function prototype should be written for each function and placed BEFORE the main procedure.
3. Each function should have a comment explaining what it does.
4. Each function parameter should have a comment explaining the parameter.

5. Declare a constant in the **main** function that stores the array size of 20. Do not declare this constant globally.

6. Implement the following functions as described below to receive full credit. Determine when to use the ***const*** specifier (see the lecture notes). The ***const*** specifier is used when a function's parameter will not be changed in the implementation of the function.

7. Write a function read_list() which prompts for the list of non-negative numbers terminated by -99. Read the non-negative integers into an array which stores integer numbers. The number of elements in the input list may be less than 20. Stop reading when either the input number is the sentinel value -99 or when 20 numbers have been read into the array. An invalid number entered will be ignored. The function takes three parameters, the array holding integers, the number of elements stored in the array (which is defined as *pass by reference*), and maximum size of the array (use the constant defined in step 5). Define the first two parameters so that they are modifiable. The function does not return any value.

8. Write a procedure print_array() which displays the values that were inserted into the array as a comma separated list and ending with a period. For example, if the user entered five integers (not including the sentinel) then only five array values are displayed (not 20). The procedure should take two parameters, the array holding integers, and the number of elements in the array. Define both parameters so that they are not modifiable. The function does not return any value.

9. Write a function find_max() which returns the maximum value found in an array of numbers. The function takes two parameters, the array holding integers, and the number of elements in the array. Define both parameters so that they are not modifiable.

10. Write a procedure array_add() which adds a number *x* to every element of an array. For instance, if the array contains (5, 3, 1) and *x* equals 2, then the procedure changes the array to (7, 5, 3). The procedure takes three parameters, the number *x*, the array, and the number of elements in the array. Define the array so that it is modifiable. The function does not return any value.

11. In the main program, use the following algorithm:
    a. Call the function read_list() to prompt and read the input list of values terminated by the sentinel value.
    b. Call the procedure print_array() to display the array values.
    c. Call the function find_max() to search for the largest value in the array.
    d. Call the procedure array_add() to add the maximum value to each element of the array. Pass the maximum value computed in the previous step as the number *x*.

   e. Call the procedure print_array() to display the array values after it was modified in the previous step.

12. Thoroughly test your program on different input. For example, the user enters no input numbers, negative values, or zeroes. Also ensure that your program does not allow the user to enter more than 20 valid numbers. Compare your output against the provided solution **`array_solution.exe`**.
13. Be sure to add the header comments “File”, “Created by”, “Creation Date” and “Synopsis” at the top of the file. Each synopsis should contain a brief description of what the program does.

14. Be sure that there is a comment documenting each variable.

15. Be sure that your *if* statements, *for* and *while* loops and blocks are properly indented.

Submit Your Work

Important: Any program which does not compile and run will receive no credit!
If you are not sure what this means please ask your instructor.

Submit the file **`array.cpp`** using the *Lab 9* drop box on Carmen. **DO NOT** submit the file **`a.out`**. **DO NOT** submit uncompleted work from *Quiz 9*. This will not be graded.