Lab 4: Released Wednesday, week 4 and due Tuesday, week 5

Note: This lab gives you all of the instructions for writing your code in Step 2, and then the instructions for compiling in Step 3. You are not required to write all three programs before compiling, feel free to write one, compile and test it, and then move on to the next.

Functions and Command-Line Arguments

Introduction

The assignment for this week will utilize concepts of arrays.

We will also be grading for meeting requirements, using “class legal” code, and plagiarism. So, it is not enough for your lab to just pass the Gradescope autograder! Please read the instructions herein carefully.

It is highly recommended that you plan the algorithms for each program first and then develop the C++ code for it.

Step 1: Getting Ready

Recall what environment you set up in lab 1. Navigate there, and then clone the Github repository. Go to the course Github (https://github.com/ucsb-cs16-1-u22) and make sure you can see a repository named Lab04-yourgithubusername. If you can see that repository, open it up in your browser and go to the green “Code” button near the upper right. Copy the SSH link for this repository (for more instructions on this step, and future steps involving Github, check out https://sites.cs.ucsb.edu/~maradowning/cs16/refs/gitbasics.html).

From here, run the following command in the folder you’d like your Lab03 folder to end up in:
`git clone <link you just copied>`

Check afterwards with `ls` to make sure the repository has been cloned properly, you should see a new folder named Lab04-yourgithubusername. You can run the following command to enter this directory and see the starter code files:
`cd Lab04-yourgithubusername`

Step 2: Edit your C++ Programs

This week, you will need to create one (1) C++ program called arrays.cpp. It is worth 100 points and submitted for full assignment credit.

IMPORTANT NOTE: We will take major points off if you use C++ instructions/libraries/code that either (a) was not covered in class, or (b) was found to be copied from outside sources (or each other).

arrays.cpp

Setup
For this project, you will start with five (5) files: ArrayFile.txt, constants.hpp, headers.hpp, arrays.cpp, and Makefile.

The first thing you should do is open and read arrays.cpp. Upon closer examination, you will notice that there are two “new” types of instructions there:

#include "headers.hpp"
#include "constants.hpp"

These lines are there to tell the compiler to go get the files headers.hpp and constants.hpp and place them in the program there. This is one way to make your programs be a little more compact. This is what each contains:

headers.hpp contains all the function declarations that you will need for arrays.cpp. There are 8 function declarations in there. You will need to create the function definitions from these:

```
void print_array(int arr[], int asize);
int maxArray(int arr[], int asize);
int minArray(int arr[], int asize);
int sumArray(int arr[], int asize);
void evensArray(int arr[], int asize);
void primesArray(int arr[], int asize);
int SeqSearch(int arr[], int asize, int target);
void AllSearches(int arr[], int asize);
```

Feel free to adjust relevant function arguments with the `const` keyword if you wish.

In addition, this file also contains a definition for a function called `getArray()`. It involves getting data from an external file and you are not responsible for doing anything with it (we haven’t learned how to do file I/O yet). DO NOT CHANGE THE `getArray()` DEFINITION AT ALL!!

The file constants.hpp contains declarations for constant global variables that you will need for arrays.cpp. Your program will be reading in a text file called ArrayFile.txt, which contains a MAXSIZE number of integers. The program assigns those integers as elements of an array called `array` via the function called `getArray()`. Again, this is a function that you do not have to define yourself—it is written for you in headers.hpp. More on this in a bit. The other 2 things in the constants.hpp file are:

- a constant int called NSEARCHES, which is set to 10.
- a constant int array called SEARCHES[] that contains NSEARCHES elements. SEARCHES[] is initialized to certain int values. Again, more on this in a bit.

Design your program

Your program `arrays.cpp` will do eight (8) tasks (they're not big tasks, individually). Each one of these 8 tasks has to do with one of the 8 function declarations mentioned earlier. There is a hint in the skeleton file for you about how to call these functions. Once your program reads the array data (via `getArray()`, which you will not modify), it should do the following things (these should all be done with 8 functions that you will have to define in your submitted file):

- Print the array that it just read in. It does this using the function print_array(), which is declared in the headers.hpp file. You are required to do this using this function only. The function call should be all it takes to execute this task.
- Find the largest integer in this array and return it. It does this using the function maxArray(), which is declared in the headers.hpp file. You are required to do this using this function only. Once this function is called, you should print out the result (see the example run below).
- Find the smallest integer in this array and return it. It does this using the function minArray(), which is declared in the headers.hpp file. You are required to do this using this function only. Once this function is called, you should print out the result (see the example run below).
- Will find the sum of all the integers in this array and return it. It does this using the function sumArray(), which is declared in the headers.hpp file. You are required to do this using this function only. Once this function is called, you should print out the result (see the example run below).
- Find all the even number integers in this array and print them. It does this using the function evensArray(), which is declared in the headers.hpp file. You are required to do this using this function only. Once this function is called, you should print out the result (see the example run below).
- Find all the prime number integers in this array and print them. It does this using the function primesArray(), which is declared in the headers.hpp file. You are required to do this using this function only. A prime number is a number that is only divisible by itself and 1. Note that 1 is, itself, not considered a prime number (2 is, however). Once this function is called, you should print out the result (see the example run below). Important note: Only print out the positive primes.

- Will search the array for every integer number in the global array SEARCHES[]. It will use both the function SeqSearch() and the function AllSearches() to do this, both of which are declared in the headers.hpp file. You are required to do this using these functions only. Here's what you have to pay mind to: SeqSearch() only finds one target in an array (the first incidence of the target). AllSearches() runs all the targets from the global array SEARCHES[].

Remember that the program must **print** out a mini-report about what it found and did not find (see the example run below).

Here is an example run using the `ArrayFile.txt` data:

```
$ ./arrays
23, 22, -5, -21, 21, 0, -12, -55, 9111, 1233, 1, 3, 5, 6, 7, 8, 9, 12, -11, 17
Max = 9111
Min = -55
Sum = 10374
Evens: 22, 0, -12, 6, 8, 12, end
Primes: 23, 3, 5, 7, 17, end
Searches:
Looking for -5. Found at index: 2
Looking for -4. Not Found!
Looking for -3. Not Found!
Looking for -2. Not Found!
Looking for -1. Not Found!
Looking for 0. Found at index: 5
Looking for 1. Found at index: 10
Looking for 2. Not Found!
Looking for 3. Found at index: 11
Looking for 4. Not Found!
```

Note that the first line is a print out of the array, then it shows the maximum, the minimum, the sum, the evens, the primes, and the searches "mini-report". The integer array is the one that is read in at the start of the program (by the getArray() function) and the array of searched numbers is the one defined in the SEARCHES[] declaration.

Please make sure your output matches the above. For example, notice how the printout of the array has commas and a space between each element, except for at the last one. Also notice that the evens and primes lists also have commas and a space between each element and they end with the word "end".

REQUIREMENT: Do not create/define any functions other than the ones that are declared in the headers.hpp file. Do not change the function declarations or modify headers.hpp (except for additions of the const keyword to arguments if you see fit).

When you test your program, it will use the ArrayFile.txt data as your source. When you want to test your program further, you can create another data file (give it another name, like MyTest.txt or something) with just integers in it (separated by whitespaces). You will have to modify the MAXSIZE and FILENAME global variables accordingly in the constants.hpp file (only modify these).

You can assume any array file given as input to your code contains at least one element.

Step 3: Compile your Code

This time you'll use a Makefile to compile your code. A Makefile already exists in this folder, with contents as follows:

```
all: arrays

arrays: arrays.cpp headers.hpp constants.hpp
	g++ -std=c++11 -Wall arrays.cpp -o arrays

clean:
	rm arrays
```

To compile your program, you can run
`make`

If you need to delete your executable file, running
`make clean`
will delete it

You can run the executable file as normal:
`./arrays`

If you encounter an error, use the compiler hints and examine the line in question. If the compiler message is not sufficient to identify the error, you can search online to see when the error occurs in general.

Remember to re-compile after you make any changes to your C++ source code.

Step 4: Submit your Code

There is a more detailed (and more general) set of instructions on the course webpage, under Quick References: https://sites.cs.ucsb.edu/~maradowning/cs16/refs/gitbasics.html. If you are new to git/github, I suggest reading through this guide as well when you submit your assignment.

Once you have finished your code (or any time you want to save your code to Github and take a break), first run `git status` to see which files you have modified/created. From there, add any files you want to end up on Github by running the following command:

```
git add <filename>
```

For this lab, most likely you will run one add command:

```
git add arrays.cpp
```

Please do not add any executable files you created by compiling your code. However, if you do end up accidentally adding them, don't worry about it for now.

Once you've added all the files you want, you will need to commit them. Run:

```
git commit -m "Your commit message here"
```

Please write a descriptive commit message within the quotes, something like "Finished Lab04" if you are done, or something more specific if you're pushing code partway through.

Finally, run:

```
git push
```

to push your changes to Github. Once you've run this command, make sure to go to your repository in your web browser and check that the updated files are there (you might have to reload).

Finally, once you're completely finished with the lab, fill out the Lab04 submission on Gradescope. It only asks you for a list of people you discussed the lab with (remember, discussion of the lab is ok but do not give classmates answers), the number of hours you spent, and has a checkbox to indicate that you're done. I will use this as an indication that your code is pushed to Github and I can begin grading.