Assignment #1 B - C Programming Basics

CMPSC 311 - Introduction to Systems Programming

Fall 2023

Prof. Syed Rafiul Hussain

Due Date: September 18, 2023 11:59 PM

In this assignment you will write simple functions in C. The purpose of this assignment is to assess your basic programming skills. You should have no difficulty in completing this assignment. If you do experience difficulty, then you should take some time to brush up on programming.

Like all assignments in this class, you are prohibited from copying any content from the Internet or discussing, sharing ideas, code, configuration, text or anything else or getting help from anyone in or outside of the class. Failure to abide by this requirement will result in dismissal from the class as described in our course syllabus.

Below are the files in this assignment and their descriptions

1. `student.h`: a header file with the declarations of the functions that you will implement.
2. `student.c`: a source file in which you will implement the functions whose declarations appear in `student.h`. In other words, you will provide the definitions of the functions declared in `student h`. (Your changes are required only in this file)
3. `reference.h`: a header file with the declarations of the functions that are identical to those defined in `student.h` except they are prefixed with reference. These are the reference functions against which your implementations will be tested.
4. `reference.o`: an object file that contains the reference implementations of the functions declared in `reference.h`. This is a binary file that contains compiled reference implementations of functions.
5. `tester.c`: unit tests for the functions that you will implement. Each unit test passes an input to your implementation of a function and to the reference implementation of the same function and compares the outputs. This file will compile into an executable, `tester.o`, which you will run to see if you pass the unit tests.
6. `Makefile` Makefile: instructions for compiling and building tester used by the make utility

Note: Although you only need to edit `student.c` for successfully completing the assignment, you can modify any file you want if it helps you in some way. When testing your submission, however, we will use the original forms of all files except `student.c`. So make sure you revert all the changes and still check all your tester pass. Do not forget to add comments to your code to explain your implementation choices.

Below are the functions you need to implement:

1. Swap - Takes a pointer to two integers and swaps the values of integers. (1 point)
2. Modify Array - Takes an array of integers and their length as input and modifies the array according to the following conditions. (1 point)
    - If the number is positive, then the element must be doubled.
    - If the number is negative, then add to the number.

    - If the number is zero, leave it unmodified.
3. Nth Fibonacci number - Takes an integer n as input and returns the nth fibonacci number. The nth term in the Fibonacci Series is defined as

```
F(0) = 0, F(1) = 1
F(n) = F(n - 1) + F(n - 2), for n > 1.
```

   (1 point).
4. Median - Takes an array and the length of the array as input and returns the median of the given array. (Length of array >=1) (1 point)
5. BubbleSort - Takes an array and their length as input and sort the array using Bubble sort(2 points)
6. Armstrong number - Takes an integer n as input and returns 1 if the number is an Armstrong number and return 0 otherwise. A number is said to be an Armstrong number if the some of each digit raised to the power of the number of digits gives the original number. For example, 371 is an Armstrong number because $371 = 3^3 + 7^3 + 1^3$. In this case, since the number of digits were 3, each digit was raised to the power of 3. For the sake of simplicity of the assignment, your assignment will only be graded with three digit numbers. (2 points)
7. Palindrome - Takes a string (character array) and the string length as input and returns 1 if the given string is a Palindrome and return 0 otherwise. (2 points)

You are encouraged to write helper functions to simplify the implementations of the above functions. You should, however, not use a library function and implement all helper functions yourself.

The sample test cases present in the `tester.c` is only for your reference in this assignment. During grading, your program will be evaluvated against multiple test cases. Each test case that passes will carry 0.2 points. Each 1 point question will have 5 test cases and each 2 point question will have 10 test cases.

To start working on the assignment, start by implementing any of the above questions in `student.c` and execute the following commands in the project directory to build and test your implementation. Do not forget to build the project after you have made any changes.

```
make clean
make
./tester
```