Assignment #1 - C Programming Basics
CMPSC311 - Introduction to Systems Programming
Summer 2022 – Dr. Suman Saha
Due date: June 10, 2022 (11:59pm) EST

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.

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`

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. `reference-m1.o (for MAC m1)`: 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.

6. `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`, which you will run to see if you pass the unit tests.

7. `Makefile`: instructions for compiling and building `tester` used by the `make` utility.

You workflow will consist of (1) implementing functions by modifying `student.c`, (2) typing `make` to build the `tester`, and (3) running `tester` to see if you pass the unit tests, and repeating these three steps until you pass all the tests. 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 usethe original forms of all files except `student.c`. Do not forget to add comments to your code to explain your implementation choices.

Below are the functions you need to implement:

1. `smallest`: Takes an array of integers and the length of the array as input, and returns the smallest integer in the array by magnitude. You can assume that the input array contains at least one element. Example : If the array contains [-4,8,9,56,-100] you have to return 4 as the smallest number by magnitude is 4.

2. `mean`: Takes an array of integers and the length of the array as input, and returns the mean of all the integers in the array.

3. `swap`: Takes a pointer to two integers and swaps the values of integers.

4. `squaredPrime:` Takes an array of integers and the length of the array as input and doubles everyprime element of the array. Example : if the input array contains [1, 7, -2], then after a call to `squaredPrime` the array will contain [1, 49, -2].

5. `sort`: Takes an array of integers and the length of the array as input and sorts the array. That is, after a call to `sort` the contents of the array should be ordered in ascending order. You can implement any sorting algorithm you want but you have to implement the algorithm yourselfyou cannot use sort functions from the standard C library.

6. `negateFibonacci:` Takes an array of integers and the length of the array as input and negates every number that comes in fibonacci series. For example if the input array contains [3, 7, 55], then after a call to `negateFibonacci` the array will contain [-3, 7, -55]. A sample for first 10 numbers in a Fibonacci series is given as [1,1,2,3,5,8,13,21,34,55].

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.