This is a group work
Due date: March 14, 2019 by 11:59 PM.
Your programs will be graded based on

- Program Specifications / Correctness
- Readability
- Documentation
- Code Efficiency and Code Elegance

Submissions:

- In your GitHub account, create a Programming_HW0
- Once you've completed the homework, you should upload your files to your Programming_HW0 repository.
- Choose one person from your group to share the link to Programming_HW0 with me for grading

Questions 1:
The Collatz conjecture concerns what happens when we take any positive integer n and apply the following algorithm:

$$n = \begin{cases} n/2, & \text{if n is even} \\ 3 \times n + 1, & \text{if n is odd} \end{cases}$$

The conjecture states that when this algorithm is continually applied, all positive integers will eventually reach 1. For example, if n = 35, the sequence is

35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1

Write a C program using the fork() system call that generates this sequence in the child process. The starting number will be provided from the command line. For example, if 8 is passed as a parameter on the command line, the child process will output 8, 4, 2, 1. Because the parent and child processes have their own copies of the data, it will be necessary for the child to output the sequence. Have the parent invoke the wait() call to wait for the child process to complete before exiting the program. Perform necessary error checking to ensure that a positive integer is passed on the command line.

Questions 2:
Write a multithreaded program that outputs prime numbers. This program should work as follows: The user will run the program and will enter a number on the command line. The program will then create a separate thread that outputs all the prime numbers less than or equal to the number entered by the user. **(You can use Java or C)**

Questions 3:
Write a multithreaded program that calculates various statistical values for a list of numbers. This program will be passed a series of numbers on the command line and will then create three separate worker threads. One thread will determine the average of the numbers, the second will determine the maximum value, and the third will determine the minimum value. For example, suppose your program is passed the integers

90 81 78 95 79 72 85

The program will report

The average value is 82
The minimum value is 72
The maximum value is 95

The variables representing the average, minimum, and maximum values will be stored globally. The worker threads will set these values, and the parent thread will output the values once the workers have exited. (We could obviously expand this program by creating additional threads that determine other statistical values, such as median and standard deviation.) **(You can use Java or C)**

Questions 4:
The Fibonacci sequence is the series of numbers 0, 1, 1, 2, 3, 5, 8, ... Formally, it can be expressed as:

$f\,ib_0 = 0$
$f\,ib_1 = 1$
$f\,ib_n = f\,ib_{n-1} + f\,ib_{n-2}$

Write a multithreaded program that generates the Fibonacci sequence. This program should work as follows: On the command line, the user will enter the number of Fibonacci numbers that the program is to generate. The program will then create a separate thread that will generate the Fibonacci numbers, placing the sequence in data that can be shared by the threads (an array is probably the most convenient data structure). When the thread finishes execution, the parent thread will output the sequence generated by the child thread. Because the parent thread cannot begin outputting the Fibonacci sequence until the child thread finishes, the parent thread will have to wait for the child thread to finish. Use the techniques described in Section 4.4 to meet this requirement. **(You can use Java or C)**