COMPSCI 3SH3 Winter, 2021
Student Name: Jatin Chowdhary
Mac ID: Chowdhaj
Student #: 400033011
Date: March 5th, 2021

Lab Assignment 3 – Threads Report

Q1 – Textbook Question 4.22

Module Description

The *stats* program is a multi-threaded – four threads – program that calculates various statistical values for an array of numbers. The statistical values calculated are average, minimum, and maximum. The values are inputted by the user at the start of the program, when the executable is run (For example: ./stats 1 2 3 4). When the program is first executed, it checks if the arguments are valid, if they are it prints a few messages, and sets the size of the array. If the argument size is not valid, the program is terminated by returning an int of 1. Next, an array is declared, and values from the command line are copied to the array. The *atoi* function is used to convert the arguments from character arrays to integers. Then, three threads are declared, each corresponding to a statistical value that it will compute. The threads are created and they execute their respective thread functions that will compute a statistical value on the int array. The return value of each thread's creation is checked for any errors. Finally, the threads finish execution and are terminated. The return value for this is also checked for errors. Last but not least, the results are printed to the terminal and the program is terminated.

There are three thread functions, each corresponding to a statistical value they compute. The thread that calculates the minimum iterates through the entire array, compares the stored value against the current value, and stores the smaller one. The thread that calculates the maximum value operates in a similar manner to the minimum thread, but instead of locating the smallest value, it searches for the greatest value. The thread that calculates the average iterates the array and sums up all the elements in an int. The final sum is divided by the number of elements in the array; the result is the average.

Note: This is a semi-high level description of the *stats* program. For more information please refer to the source file *stats.c*, and use the *Makefile* to compile it.

***Source Code:** *stats.c*, *Makefile*

Q2 – Textbook Question 4.23

Module Description

The *primes* program is a multi-threaded – two threads – program that calculates prime numbers up to *n*, where *n* is a number specified by the user in the command line argument. Prime numbers are computed on a separate thread. The program starts off by checking the number of arguments given in the command line. If there are not enough or too many arguments, then the program will terminate with a return value of 1. Then, the program converts the second argument in the command line argument array to an int using the *atoi* function. The newly converted int is checked to see if it is less than 2. If so, the program terminates with a return value of 1. Next, the thread for computing the primes is declared, and then it is created and executed. The thread creation process is checked to see if it fails, and if it does an error message is printed. On the other hand, if the thread creation process proceeds normally, then the newly created thread computes all the prime numbers up to *n*, and prints them to the terminal. Finally, the thread is terminated, and the main function ends with a return value of 0.

The thread that is responsible for calculating prime numbers up to *n* does its job by iterating from 2 to *n*, and checking if each number is a prime. Each number's primality is checked by taking its square root and then iterating from 2 to the square root, and checking if these numbers can be factored. For more information on my implementation of finding prime numbers, please refer to the source code and comments located in the source file.

Note: This is a semi-high level description of the *primes* program. For more information please refer to the source file *primes.c*, and use the *Makefile* to compile it.

***Source Code:** *primes.c*, *Makefile*

**Source Code is attached in a zip file*