EE 4374 Operating Systems Design
Programming Assignment #3: Multi-threaded Prime Number Search
Due Date: April 6, 2017 (before Midnight)

Objectives:

1) To learn how to create multi-threaded programs using the POSIX pthreads library.
2) To learn how to provide user interactivity in a multi-threaded program.
3) To learn how to balance the load among threads to provide the highest speedup.

Tasks:

1) Unpack the Programming Assignment 3 template provided by the instructor into your home directory: ‘tar zxvf student_prog3.tgz’. This will create a directory called ‘student_prog3’, please rename this directory to firstinitiallastname_prog3 using the ‘mv’ command. For example, the instructor would rename the directory by executing ‘mv student_prog3 mmcgarry_prog3’. Next, go into the directory and rename all of the files from ‘student_*’ to ‘firstinitiallastname_*’ much like you renamed the directory.

2) Write a test_prime() function that returns a 0 if the integer argument is not a prime number, 1 if it is prime.

```
int test_prime(int n);
```

3) Write a prime_search() function that serves as a start routine for your prime search threads. This function will be given a range of integers and determine which integers in that range are prime. Prime numbers will be printed to a file “primesx”, where x is the thread number. Each prime number will be on a separate line in the file.

```
void *prime_search(void *param);
```

4) Write a mini_shell() function that serves as a start routine for the interactivity thread. This function will display a prompt and take commands from the user.

The following single character commands must be supported:

‘1’ : will return the integer prime search thread 1 is currently checking
‘2’ : will return the integer prime search thread 2 is currently checking
‘a’ : will return the integers both prime search thread 1 and 2 are currently checking

```
void *mini_shell(void *param)
```

5) Write a main() program that searches the first **five million** integers for prime numbers by using two threads. A third thread is created to enable the user to

check the search status of the two prime search threads while they are running. When the prime number search is completed the main function will combine the individual files “primes1” and “primes2” into a single output file “primes”. Implement test_prime(), prime_search(), and mini_shell() as library functions in firstinitiallastname_prime.c/h and main() in firstinitiallastname_lab3.c.

6) Balance the search load among the two search threads to decrease the execution time as much as you can.

7) Use a Makefile to build your program.

8) Submit the deliverables, indicated below, as a single tarball file named firstinitiallastname_prog3.tgz through Blackboard.

Deliverables:

1) Submit all of the source files in your Programming Assignment 3 directory as a single tarball file. You can create this by changing to the directory above your Programming Assignment 3 directory and execute ‘tar zcvf firstinitiallastname_prog3.tgz firstinitiallastname_prog3’. As an example, the instructor would execute ‘tar zcvf mmcgarry_prog3.tgz mmcgarry_prog3’.

Scoring:

Operation/Successful Demonstration	**65%**
Does the program build without errors or warnings using a Makefile? 20%	
Does the program find the desired range of prime numbers? 10%	
Is the program multi-threaded as described? 30%	
Has the load among the threads been optimally balanced? 5%	
Adherence to Specifications	**25%**
Does your program use the function prototypes specified in the assignment? 15%	
Does your submission adhere to the filename guidelines? 10%	
Code Readability	**10%**

Is the source code well-documented (file header comments, function header comments, comments describing large basic blocks)? 10%	
Lateness	**-10%** per day (including weekends and holidays)