Source: http://www.csce.uark.edu/~aapon/courses/concurrent/assignments/homework2.html
Modified to be the **practice assignment 3**

For this assignment you are to write a program in either C or C++ using the pthreads thread library.

Pthread Program

The goal of this assignment is learn basic thread programming. You will use mutex locks to avoid race conditions on data shared by several threads. You are required to write a C or C++ program using the pthread library as follows:

The program accepts two types of command line parameters. Valid command line executions of the program include:

```
ParallelPrimes (print a usage message and exit)
ParallelPrimes number (execute the program with a default of 2 threads)
ParallelPrimes number numthreads (execute the program with numthreads)
```

The first parameter, number, is the largest number that the program will test for primality. The program will test all numbers from 2 up to the entered number. If number is not specified then the program should print a usage message and exit. The numthreads parameter gives the number of threads that should be created. Use a default value of 2 for the number of theads that are created. Note: a number is prime if it is not divisible by any numbers other than 1 and itself. For example, numbers up to 10 that are prime include 2, 3, 5, and 7.

- The main procedure in ParallelPrimes is the master thread. The master creates some number of slave threads. Each slave thread locks a mutex and "takes" the next untaken number to work on to see if prime. (Hint: maintain a single global variable that is the index to a vector of numbers, 0 up to number.) When done, the slave thread marks the number as prime and takes a new number to test it for primality. If all numbers have been taken then the thread exits. Make sure that two threads don't test the same number for primality. The master thread waits for all slave threads to complete by executing a join. After all slave threads have completed the master thread prints the results of the primality testing.

  Note that this program is not very efficient.

  Remember to do error checking on the command line arguments. If the parameters to the program are invalid, print an informative message and exit.

Test input data:

1. `ParallelPrimes`
   Output: Usage: ParallelPrimes number numthreads
2. `ParallelPrimes 20 3`
   Output: Primes found: 2, 3, 5, 7, 11, 13, 17, 19
3. `ParallelPrimes 20 20`
   Output: Primes found: 2, 3, 5, 7, 11, 13, 17, 19
4. `ParallelPrimes 20 30`
   Output: Primes found: 2, 3, 5, 7, 11, 13, 17, 19
5. `ParallelPrimes 1000 4`
6. `ParallelPrimes 1 2`
   Output: Error: number must be at least 2.
7. `ParallelPrimes abc`
   Output: Error: number must be an integer.

Useful Resources

Apart from the usual man pages, you may find the following web sites useful:

- Linux Threads Frequently Asked Questions (FAQ) [http://www.tldp.org/FAQ/Threads-FAQ/index.html]
- Getting Started With POSIX Threads [http://www.thinkbrown.com/programming/pthreads.pdf]
- Linux Tutorial: POSIX Threads [http://yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html]