CS530 Assignment #2

Purpose

This assignment is designed to familiarize you with Pthreads.

Requirements

This assignment consists of five major requirements.

1) Developing solutions to the problems below.
2) Ensuring that your programs compile and run correctly using the tools available in a standard linux distribution.
3) Documenting your solutions correctly.
4) Organizing your *git repository* correctly.

Problem

1) In total your project will implement a multithreaded pseudoprime number finder. To do this we will be using a bignumber library for C called GMP[1]: https://gmplib.org/. You can install GMP by using apt-get in a debian distribution: `sudo apt-get install libgmp3-dev`.

   To better understand pseudoprime numbers please check the ever-benevolent overlord of information. https://lmgtfy.com/?q=pseudoprime

   To accomplish this we will implement three parts, a thread safe linked list that will allow us to shuttle data between threads easily. A thread safe number generator. A pseudoprime number checker (we will be using Miller-Rabin). And finally, some control logic that will organize the actions of the other threads.

   a. Part 1 will be implemented in C. prog2_1.c & prog2_1.h will be a thread safe linked list implantation. Your header file should contain the following struct and function prototypes:

```
typedef struct tsafenode {
      struct tsafelist *next;
      mpz_t number;
} TSAFENODE;
typedef struct tsafelist {
      pthread_mutex_t *mutex;
```

[1] https://www.cs.colorado.edu/~srirams/courses/csci2824-spr14/gmpTutorial.html

```
        TSAFENODE *head;
} TSAFELIST;
typdef struct tsafereturndata {
        // True(1)/False(0) if returned data is valid.
        int isValid;
        mpz_t value;
} TSAFEDATA;
TSAFELIST* tSafeConstruct();
void tSafeDestruct(TSAFELIST*);
void tSafeEnqueue(TSAFELIST *,mpz_t);
TSAFEDATA tSafeDequeue(TSAFELIST *);
```

The functions should implement a first in first out linked list (AKA a queue). The `tSafeConstruct` function should allocate memory for a `TSAFELIST` on the heap. The `tSafeDestruct` function should free any associated memory. The `tSafeEnqueue` function should add an element to the end of the list. The `tSafeDequeue` function should remove the front of the list and return it, if there is no element on the front of the list it should return the TSAFEDATA element with isValid set to false.

b. Part 2. It is simplest to implement the rest of the project as a single C file. prog2_2.c will implement the rest of the behavior. To do this I recommend first by implementing the following global variables:

```
mpz_t globalCounter;
pthread_mutex_t counterguard = PTHREAD_MUTEX_INITIALIZER;
TSAFELIST *numberList;
```

Your program should start up by initializing the globalCounter & numberList.

You should also write a function that will generate the next number to check in a thread safe manner.

```
mpz_t getNextNum();
```

Next, I recommend that you implement the pseudoprime number generator function.

```
void * generatePrimes(void *);
```

This function, in a loop, will get a number to check, run Miller-Rabin checker for 100000 iterations. If the number is pseudoprime enqueue it on the numberList. **MASSIVELY HUGE HINT:** `mpz_probab_prime_p()`

Next I recommend that you write your main program. Your program will take in a number K and a bit length B as the first and second command line arguments

respectively. Your program will then find K pseudoprime numbers of at least bitlength B^2.

That is, if K = 10 and B = 1024, your program will generate and print to STDOUT any 10 pseudoprime numbers that are of at least bitlength 1024 bits. Each of the pseudoprime numbers should be printed on their own lines. They should each be printed in base 10.

Your program should do this by starting four threads that's will be running the generatePrimes function. Your main thread can then collect the pseudoprime numbers from the threads by repeatedly dequeuing values and collecting them until it has collected K numbers that fit the bitlength requirements. You can print the numbers as you find them. It would be smart to have your main thread sleep between dequeue attempts so that it doesn't busy wait and starve the worker threads.

2) You must ensure that your code can be run on a standard Debian based linux distribution using the shell tools of your choice. You may develop your solution on any machine you desire, as long as the final solution works on a standard linux distrubtion.
3) Your solution must have a complete comment header as is detailed below. During runtime, each of your solutions to section 1 must output a correct **title string** as the first line printed. It should be in this format:

```
Assignment #1-1, <NAME>, <EMAIL>
```

Each problem should follow this format, with the second number incremented for each problem in part 1. For example the second problem in part one should have the title string:

```
Assignment #1-2, Scott Lindeneau, slindeneau@gmail.com
```

4) You must place a copy of your solutions inside a repository named cs530Assignment2 in your git profile on gitlab.
You will also place a README.md file[3] in your assignment directory. This file should contain a description **IN YOUR OWN WORDS** of the project along with a short description of each file, what it is, how to compile and/or run it. These descriptions may be short as long as they are accurate.

YOU MAY NOT HAVE ANY OTHER FILES IN YOUR GIT REPOSITORY. If you have any other files in your git repository the assignment will be considered **not** complete.

[2] https://stackoverflow.com/questions/7150035/calculating-bits-required-to-store-decimal-number

Late Policy

Programs turned in by the due date will receive 120%, programs turned in 7 days late will be worth 100%, after 7 days 0%.

Cheating Policy

There is a zero tolerance policy on cheating in this course. You are expected to complete all programming assignments on your own. Collaboration with other students in the course is not permitted. You may discuss ideas or solutions in general terms with other students, but you must not exchange code. (Remember that you can get help from me. This is not cheating, but is in fact encouraged.) I will examine your code carefully. Anyone caught cheating on a programming assignment or on an exam will receive an "F" in the course, and a referral to Judicial Procedures.