Lab 6: Multithreading

Start Assignment

- Due Oct 6 by 3:15pm
- Points 10
- Submitting a file upload
- Available after Aug 26 at 12am

Problem 1: Multithreaded Statistical Calculation

Based on the Pthread library, write a multithreaded program.

This program should first get two integer numbers (e.g., from the command line or simply hardcoded as an array) and then create (***pthread_create***) three separate "worker" threads:

- one thread will determine the average of these two numbers,
- the second will determine the larger one of them, and
- the third will determine the smaller one.

For example, suppose your program is passed from the command line with the following integers,

```
89 91
```

The program will report

```
The average value is 90
The larger value is 91
The smaller value is 89
```

The variables representing the average, larger, and smaller values could be stored globally.

The "worker" threads will set these three values, and the parent thread will output these values when all worker threads have exited (***pthread_join***).

(*We could obviously expand this program by getting more than two numbers from input and creating additional threads that determine other statistical values, such as median and standard deviation.*)

Use the Pthreads library to implement this program (link with -lpthread if needed).

Problem 2: Multithreaded Fibonacci Series Generation

The Fibonacci sequence is the series of numbers

```
0, 1, 1, 2, 3, 5, 8, ...
```

Formally, it can be expressed as:

```
f(0) = 0
f(1) = 1
f(n) = f(n-1) + f(n-2)
```

Write a multithreaded program that generates the Fibonacci sequence.

This program should work as follows:

From the command line, ask the user to enter the number of Fibonacci numbers that the program is to generate (this number should not be too large, why?).

The program will then create (***pthread_create***) a separate thread (i.e., the "worker" thread) that generates the Fibonacci numbers.

The worker thread places the Fibonacci sequence in data variables that can be shared by the threads (an array is probably the most convenient data structure).

When the worker thread finishes, the parent thread should output the sequence generated by the child (worker) 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 (***pthread_join***).

Use the Pthreads library to implement this program (link with -lpthread if needed).

Submissions:

- source code files
- test results screenshots