CSE 2431/5431 **LAB 3 (**)**

1. Goal

Understand how to create threads and use locks.

2. Creating threads

In the class, I have shown that creating threads is actually tricky. Please read slides 15-19 in concurrency.pdf. I have provided the faulty program as createThreads.c. You need to fix it with the malloc approach.

3. Using locks

First take a look at the provided sequential.c. It generates a million random integers between 0 and 100 and then calculates the sum of these integers. Compile it and run it. Note that we provide a deterministic seed to the random generator, so every time you run it, you should get the same result.

Experiment 1: we hope to parallelize the computation by creating 10 threads and let each thread sum 100,000 numbers. Take a look at parallel1.c. It creates 10 separate arrays, each holding 100,000 numbers. You need to add code to create threads, pass one array to each thread, and let each thread perform its work. **In Experiment 1, let's try it without synchronization.** Each thread should do the following work:

```
for(i=……){
    sum += array[i];
}
```

Compile and run your program. Does it generate the same result as the sequential version? Run it multiple times. Does it always generate the same result?

Experiment 2: now let's add proper synchronization to your code. You need to define a lock in your program and change each thread's code to the following:

```
for(i=……){
    lock()
    sum += array[i];
    unlock()
}
```

Put your code in parallel2.c. Compile and run your program. Does it generate the same result as the sequential version? Run it multiple times. Does it always generate the same result?

Experiment 3: the previous program needs to lock and unlock many times. Could you think of a better solution that does not need to hold the lock for multiple times or for a long time? Put your code in parallel3.c

4. Testing

For experiments 2 and 3, the result of a correct multi-threaded version should always generate the same result as the sequential version.
Since a faulty multi-threaded program may just occasionally show a wrong result, you need to run your program multiple times.

5. Submission

Submit createThreads.c, parallel1.c, parallel2.c, and parallel3.c, and Makefile in a zip file on Carmen. Your Makefile should generate four output files called createThreads, parallel1, parallel2, and parallel3.