NAME: Raymond Shum
DATE: April 13, 2021
TITLE: Lab 6 – Part 4 – Write-Up
DESCRIPTION: This answers the question found in Step 1 in the lab.

Explain what happens when you run the threadSync.c program?

```
Thread 0 Entered Critical Section..
Thread 1 Entered Critical Section..
Thread 0 returned
Thread 2 Entered Critical Section..
Thread 1 returned
Thread 3 Entered Critical Section..
Thread 2 returned
Thread 4 Entered Critical Section..
Thread 3 returned
Thread 5 Entered Critical Section..
Thread 4 returned
Thread 6 Entered Critical Section..
Thread 5 returned
Thread 7 Entered Critical Section..
Thread 6 returned
Thread 8 Entered Critical Section..
Thread 7 returned
Thread 9 Entered Critical Section..
Thread 8 returned
Thread 9 returned
Main thread done.
```

The threadSync.c program creates 10 threads. Each thread runs *go as its starting function. Each thread must acquire the lock (in the form of the mutex semaphore) before it proceeds through the critical section in go. Once it acquires the lock, it prints the output seen in the screenshot before releasing it.

The mutex semaphore (initialized to 1) only allows one thread to proceed through the critical section at a time. This is because the value of the mutex semaphore is decremented to 0 once the first thread acquires the lock. When a second thread calls sem_wait(&mutex), it decrements its value to -1, which puts the thread to sleep. To release the lock, the first thread calls sem_post(&mutex), which increments the value back to 0 and awakens another thread. This allows the next thread to proceed.