Semaphore Project Summary

This project used semaphores in order to create order and mutual exclusion of critical sections in a program that simulated a DMV workday. The biggest difficulty I faced was in the cleanup of threads at the end. When I signaled each thread to end, they were all stuck in an infinite loop because they were waiting for semaphores to be signaled. In order to remedy this issue, I needed to signal each semaphore once in the method that ended each thread.

In these methods, I set a boolean variable to "false" so that the while loop in the run() method did not loop again. The problem with this was that the threads were already waiting for the first semaphore in the beginning of the while loop when I changed the while variable to false. Therefore, after my signalStop() method set the boolean to false, I modified it to then signal each semaphore in the run() method. This made it so that the thread would be signaled all the way through and encounter the while(false) at the beginning of its next loop. Once it encountered this, it skipped all the code and then exited.

Another issue I encountered was after these threads were being pushed through all their semaphores, they ended up printing to the screen for their activities, even though they weren't actually doing anything. In order to cleanup the output of the program, I added an If() to each print statement that only kept the thread from printing when the thread's while() loop was already set to false. Although it made the code look more unappealing, it definitely helped with the style of the output and terminal screen.