# The City College of New York
# Computer Science Department
# Operating Systems
# CSC 33500
# Lab 6
# By Nafis Khan

**Introduction:**

This lab contains two different ways of achieving the same result. One uses pthreads and the other uses semaphores to create an environment where an agent supplies two smoking ingredients at a time on the table and three smokers pick them up individually, to create a cigarette and smoke it.

**POSIX Thread Program:**

The pthread program runs synchronously using multi-threading to complete this job. Initially, pthread mutex locks are created so that synchronicity could be maintained when different threads are called. Inside the main function, threads are created and assigned functions in the order of agent, smoker 1, smoker 2, and smoker 3. In the functions, the threads are locked and unlocked as needed so that all the threads work in the correct order. This prints the actions of the agent first, and then the smokers. The agent provides ingredients x (10 times from prompt in this case) number of times. When all operations are completed, the agent locks its own thread and cancels the other ones so that the program stops right after and no other action from the agent or the smokers is taken since the smokers keep taking from the table unless the threads are stopped some way.

**Semaphore Program:**

The semaphore program uses C programs semaphore functionality via the sem.h file provided. It creates all the semaphore variables needed and then goes on to create children processes using fork to deal with the agents and smokers. In total, four children processes are created, the first one being the agent and the rest of them being the smokers. The Agent operations run on a while loop of x number of times just like the other program and so can repeat multiple instances of putting random ingredients on the table. When the Agent decides what to put on the table, an appropriate semaphore for the smoker who has the rest of the ingredients is unlocked right there, and therefore the smoker in one of the nested child processes is only able to pick it up. The smoker who just picked up the ingredients and smoked then locks the appropriate semaphore again. This process of unlocking the correct semaphore for the correct smoker repeats and locking results in the same output as the other program but using a different method.