Task 6 – Report

William Lee

For this lab assignment I created two code one with semaphores and one with pthread. For the semaphore implementation I created five semaphores, one for each smoker, the agent, as well as a general semaphore called lock. The program then forks a child process for each smoker and the agent will run in the parent process. Each smoker process has the same implementation so I will be explaining smoker_paper process. The smoker_paper process runs in an infinite loop where it first tries to wait for its own semaphore. Note that all semaphores are initialized to zero except for the lock semaphore which means that all the semaphores are waiting for a signal. If the smoker_paper process can wait for its own semaphore (meaning that the semaphore has a value of one) then it will wait for the general lock semaphore. This general lock ensures that only one process is running at a time and that there is no process that runs their content concurrently which may cause data to be overwritten. If the smoker_paper process can obtain the wait for lock semaphore it prints out a message stating its actions and signals the agent semaphore and signals the lock semaphore so that other processes can wait for the lock. Next is the agent process which runs in a for loop for ten iterations in this case. It first waits for the lock semaphore and then generates a random number from one to three. Based on the random number generated it will signal the appropriate smoker semaphore. Note that due to all the smoker semaphore to initialize to zero, the smoker processes can’t run until the agent runs and signals one of the smoker semaphores. This helps ensure that at the agent process is always the first process to run and follows up by the appropriate smoker process. After signaling the smoker semaphore the agent proceeds to signal the lock semaphore and wait for the agent semaphore, putting it to sleep until the smoker process signals the agent process as explained above.

For the pthread implementation of the assignment it’s similar. Instead of using semaphores, mutexes are used to lock the critical section of the code and instead of processes pthreads are used to run the smokers and agent. Five mutexes are created, one for each smoker, the agent, and the general lock. Four pthreads are created first being the agent, then the smokers. Each thread is passed their respective void function to run, for example the agent thread will be passed the agent function to generate random numbers and same is applied to the smoker threads. After the threads are created, pthread_join is used to wait for the agent thread to finish as its program is based on how many times the agent can place items on the table. Like the semaphore implementation the agent thread always goes first. Inside the agent function it locks all smoker mutexes and then locks the agent mutex and then locks the general lock mutex. Then it continues to generate a random number ranging from one to three and unlocks one of the smokers mutex. This allow only one of the smoker threads to run as the rest of the smoker thread are waiting for their mutex to be unlocked. The agent thread then unlocks the general lock mutex and locks its own mutex to ensure that the agent can’t go again after unlocking an agent. The smoker thread functions are the same to the process in the semaphore implementation as it tries to lock its smoker mutex and then continues locking the general lock before printing out its actions and unlocking the agent mutex that the agent locked.