**COMP 310/ECSE 437**

**Assignment 3 – Part 1**

**Theoretical Questions**

1. *When we already use 2 semaphores in the producer-consumer problem, why is there a need for mutex?*

Once locked, a mutex can only be unlocked by the thread that locked it, whereas semaphores can be signalled by any other thread. The mutex is necessary since we have more than one producer and consumer threads. If 2 producers are trying to add to the buffer, it might lead to race conditions, as both are fighting for the same resource in the critical section. Having a mutex ensures that only one thread is adding or removing from the buffer at a time.

1. *Is it possible that a consumer with lowest priority suffers from starvation in the 2 semaphores and 1 mutex setup for producer-consumer? Explain the situation.*

Yes, it is not possible for a consumer with lowest priority to suffer from starvation. For instance, imagine that the buffer is emptied by high priority consumer threads, and there are no more producer threads to fill it back up. Thus, the lower priority consumer is waiting for their turn, but it will starve unless a new producer fills the buffer.

1. *Though binary semaphores avoid starvation and mutexes don’t, why is it recommended to use mutex in producer-consumer to secure the critical section?*

Mutexes are a way to ensure that there is only ever one thread accessing critical section at a time, thus preventing race conditions. It is also beneficial in the sense that only the thread that locked can unlock, which is not the case for binary semaphores. Binary semaphores allow another thread to “unlock” and start running. Therefore, it is recommended to use mutex, since we want the same thread to be in control, and release when it is done the entire critical section, not have another thread interrupt it.

1. *Why do producer-consumer need to have 2 semaphores namely “Full” and “empty”. What complications may arise if we use only one semaphore for “Full” and rely on computed complementary value for “empty”.*

If we only use one Full semaphore, which represents the number of items in the buffer, then there is no way that the producer can be directly alerted that the buffer is full, and thus stop it from running. The only way for the producer to know it is full would be for it to constantly check the value of Empty, representing the number of available spaces in the buffer, thus leading to busy waiting. Busy waiting takes up CPU time.