David Lee

0028130595

Lab 1 Analysis

1. My solution guarantees that only one process will attempt to delete an item at a given time by using a semaphore to work as a mutex for mutual exclusion. The semaphore is initialized with a value of 1 so that if a consumer or producer is making changes to the buffer, nothing else can make changes to the buffer at the same time.
2. Producers and consumers cannot concurrently access the shared buffer, since they could be pointing to the same location. I made sure of this by making the mutex semaphore be used for both producers and consumers so that only one process is accessing the shared buffer at the same time. We don't want the shared buffer to be changed at the same time because the item that is currently removed may be changed in the same process.
3. When a consumer starts to extract items, the system will guarantee that the consumer will receive contiguous locations. This is because I have a semaphore acting so that it tells me whether the shared buffer is full or empty, so that we don't get values outside of the buffer as well as two pointer pointing to indices where the shared buffer should be adding and removing values.
4. Extra credit
    a. When the producers have a higher priority than the consumers, I had the producers create input more items into the shared buffer than the consumers removed.
    b. When the consumers have a higher priority than the producers, I didn't see much of a difference compared to when they were of the same priority, but I would assume that is because when the priorities are the same the number produced should be like the number consumed.
    c. When each consumer has a unique priority value, the consumer with the higher priority had removed the most items.
    d. When each producer has a unique priority value, the producer with the higher priority had input the most items.