Name: Sanjit Sama

1. (5 pts) How does your solution guarantee that only one process will attempt to delete an item at a given time? My solution guarantees that only one process will attempt to delete an item at a given time through the use of semaphores and utilizing the wait() and signal() commands. Within the consumer method the critical section of my code is protected through the use of wait() and signal(). Essentially I have two process types of produced and consumed. The consumer methods waits for the produced processes and then signals the consumed process. This happens within the remove_buffer method. The locked semaphore protects my critical section.
2. (5 pts) In your solution, when a producer needs to insert an item in the shared buffer, are consumers also excluded, or can consumers continue concurrently? Explain. In my solution, consumers and producers cannot concurrently access the shared buffer. This is because the shared buffer is considered to be a critical section since modification of the buffer simultaneously by multiple process may corrupt the data. Hence, as previously mentioned the wait() and signal() commands act like a locking mechanism. They surround the critical section in this case the shared buffer protecting from being shared.
3. (5 pts) When a consumer starts to extract items, does your system guarantee that the consumer will receive contiguous locations in the shared buffer, or do consumers contend for items? Explain. No they do not contend for the items. Reason being, the semaphores and mutex locks only allow for one process at a time.
4. Your results from the extra credit experiment if you completed it.

Extra Credit

In your initial implementation, all processes are created with a priority of 20. Experiment with changing the process priorities:

- The producers have higher priority than the consumers.
- The consumers have higher priority than the producers.
- Each consumer has a unique priority value.
- Each producer has a unique priority value.

After running the program with a higher priority for producers than consumers I consistently receive one extra produced item. Rather than 90 I receive 91. This is due to the fact that there is a higher chance that the scheduler may give priority to the producer process over the main allowing for it to get one more item created rather than settling for the main process which would shut down the program and stop the simulation.

When the consumer is higher we notice that the number of items created returns to 90 and that there is a more uneven distribution of items consumed by each producers. So some are consuming 30 or 29 while others are 0 or 1.

When there is a unique priority value for all the producers I notice that all of the producers were able to produce 1 more item. The way I tested this was by incrementing the producer priority by 10 for each process starting from 20.

With a unique consumer priority and producer priority set at 20 we see a more balanced consumption from a consumer standpoint and where it deletes the item from. In the past we've seen uneven distribution of some deleting 0 from certain producers and others deleting up to 31. Here we see more of a balance and closer to 10 deleted items.

All in all, to answer the underlying question if this all considered to be fair there are two different lens that we may view this issue from. If we consider fair to be in a purely technical, behind the scenes sense, we most definitely can consider this unfair since inevitable the priorities are skewed allowing greater importance from the scheduler. However, that being said, when we view this from a pure output sense I do not think this is unfair. I understand that we are not working with many process and or is this a very long program with many process, hence, our outputs are quite similar. The variety of iterations listed below do not vastly deviate from the norm. I'm inclined to say that this is most definitely fair if we are considering from an output sense. I think this idea would be backed by those also not having a technical understanding of what is happening. If we showed these results to others without the technical knowledge I'm inclined to say that they would not believe this outputs are vastly different and would believe for it to be "fair".

Normal
Producer A: created 30 items
Producer B: created 30 items
Producer C: created 30 items

Consumer a: deleted 1 items from producer A, 2 items from producer B, 27 items from producer C
Consumer b: deleted 27 items from producer A, 1 items from producer B, 3 items from producer C
Consumer c: deleted 2 items from producer A, 27 items from producer B, 0 items from producer C

Producer Higher:
Producer A: created 31 items
Producer B: created 30 items
Producer C: created 30 items

Consumer a: deleted 1 items from producer A, 2 items from producer B, 27 items from producer C
Consumer b: deleted 27 items from producer A, 1 items from producer B, 3 items from producer C
Consumer c: deleted 2 items from producer A, 27 items from producer B, 0 items from producer C

The shared buffer contains: 1 items from producer A, 0 items from producer B, 0 items from producer C

Consumer Higher:
Producer A: created 30 items
Producer B: created 30 items
Producer C: created 30 items

Consumer a: deleted 1 items from producer A, 0 items from producer B, 29 items from producer C
Consumer b: deleted 0 items from producer A, 30 items from producer B, 1 items from producer C
Consumer c: deleted 29 items from producer A, 0 items from producer B, 0 items from producer C

The shared buffer contains: 0 items from producer A, 0 items from producer B, 0 items from producer C

Unique Producer Priority
Producer A: created 31 items
Producer B: created 31 items
Producer C: created 31 items

Consumer a: deleted 1 items from producer A, 30 items from producer B, 0 items from producer C
Consumer b: deleted 30 items from producer A, 1 items from producer B, 1 items from producer C
Consumer c: deleted 0 items from producer A, 0 items from producer B, 30 items from producer C

The shared buffer contains: 0 items from producer A, 0 items from producer B, 0 items from producer C

Unique Consumer Priority
Producer A: created 30 items
Producer B: created 30 items
Producer C: created 30 items

Consumer a: deleted 12 items from producer A, 0 items from producer B, 6 items from producer C
Consumer b: deleted 12 items from producer A, 13 items from producer B, 18 items from producer C

Consumer c: deleted 6 items from producer A, 17 items from producer B, 6 items from producer C

The shared buffer contains: 0 items from producer A, 0 items from producer B, 0 items from producer C