Briefly describe the problem you are trying to solve and describe the design of your solution.

We are trying to implement producers and consumers that produce and consume items using a queue with limited numbers of items. Every item produced must be consumed, and the implementation must be “thread-safe” since the producer and consumer need to be able to be their own processes and we can have multiple producers and consumer processes. We also have the issue of having a “bounded buffer”. This means that there can only be so many objects on the queue waiting to be consumed at any given time.

The solution is to make sure that the producer(s) and consumer(s) cannot access the queue concurrently using a mutex, but also checking if the buffer “isFull” or “isEmpty” (producer and consumer respectively) and if they can’t do anything with the buffer at that moment, waiting until the signal is fired that the buffer is “notFull” or “notEmpty”. The “notEmpty” signal is sent after the producer puts something on the queue, and the “notFull” signal is sent after the consumer consumes an object from the queue.

The final signals are sent as broadcasts so any waiting threads will wake up and attempt to access the queue again, however some may still find the queue full or empty since the other processes may have already either filled the queue (producers) or emptied the queue (consumers) before the process got to take its turn. This broadcast makes sure that all producers and consumers eventually wake up and exit. This method allows the producer(s) and consumer(s) to wait until they can do something without utilizing excessive amounts of unnecessary CPU resources and without having race conditions.

Discuss why the busy-wait solution is inefficient for threads running on the same CPU, even though it produces the "desired behavior".

The busy-wait solution uses CPU resources by repetitively checking the conditions. These resources are better utilized by other threads that are doing useful things (adding to or removing from the buffer). Just checking the same conditions (if the buffer is empty or if the buffer is full) over and over is not an efficient use of resources.

You will need to argue from your narrated output as to why your solution is correct. Note, your output will likely not match the output listed here exactly. Two successive runs of your application will probably not match even vaguely, due to random variations in how threads are scheduled. However, your report

should discuss each of the following points and discuss how your output supports your discussion of each:
Are each producer and consumer operating on a unique item? Is each item both produced and consumed?
Do the producers produce the correct number of items and the consumers consume the correct number of items?
Does each thread exit appropriately?
Does your solution pass the above tests with the different numbers of producer and consumer threads in the Makfile tests?

In all the test cases there are 30 items produced, numbered from 0 to 29 in the outputs. In all test cases all items from 0 to 29 are both produced and consumed (they are always produced before they are consumed) and are only produced and consumed once. If we have produced 30 items, the producers will exit. Similarly, if we have consumed 30 items, the consumers will exit. If the queue is full the producer will output FULL and wait until the queue is no longer full before producing a new item. If the queue is empty when a consumer sees it, it will output EMPTY and wait until the queue is no longer empty before checking the queue again to see if there is an item to consume.

Each thread completes if the total produced is the maximum number that we want to produce, then the producer exits. The producer threads will all have their own exit statements. The consumers work the same way. If the total number consumed is equal to the maximum number then it exits the loop and outputs “exited”. Each thread will always give an “exited” output at the end of its use suggesting that all the threads are exiting properly.

My solution passes all of the tests listed above regardless of the number of producers and consumers. Also, all the makefile tests pass the above tests.