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

We are trying to control concurrency to ensure correct computation behavior for the producer-consumer problem. The solution is using both semaphore and mutex. Mutex is for blocking the critical section being accessed by other threads. In producer function, we wait until the queue is emptied by the consumer after done producing, then do the critical section with the mutex. In the consumer function, we wait until the queue is full by the producer after adding to the queue, then do the critical section with the mutex, and do consuming work outside the critical section.

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

Even if the thread avoids busy waiting, we are still wasting a lot of cycles by forcing each thread to do nothing when the condition required for them to continue is not met.

3). Why are you confident your solution is correct? 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, you should be able to discuss each of the following points and discuss how your output supports your discussion of each:

a). Are each producer and consumer operating on a unique item? Is each item both produced and consumed?

No, the producer produces the item and the consumer consumes it. Yes, each item produced by the producer will be consumed by the consumer eventually.

b). Do the producers produce the correct number of items and the consumers consume the correct number of items?

Yea, both reached a maximum of 30 times.

c). Does each thread exit appropriately?

Yes, I would assume so. Because the sorted file all ends up with an exited message

d). Does your solution pass the above tests with the different numbers of producer and consumer threads in the Makefile tests?

Yes, all tests run without deadlocks.