Q & A related to Assignment 2

1. Is it normal to have multiple customers to arrive at the same time?

Answer: Yes, it is possible and it is normal.

2. As stated in the rubric we must handle Special cases (illegal values), since the file must be in proper format what illegal values could we expect and what should we do to handle them?

Answer: You need to catch the illegal values in the input file. To be more specific, check if the arrival time is a positive integer and if the service time is a positive integer. To simplify, you do not need to check the illegal values for other fields (e.g., the uniqueness of customer id, and so on).

3. Do we need to destroy mutexes, some mutexes are owned by the clerks and therefore the main thread fails when destroying it since it doesn't own these mutexes. How should we handle this?
Answer: While it is a good practice to destroy the mutex after its lifecycle, it is not mandatory. In your case, you can make mutex global variables so that you can destroy it in the main thread.

4. Do I need to catch all system call return values?
Ans: While *in principle* you should catch all possible exceptions, *in practice, nobody did that.* In practice, people catch exceptions based on the importance of the exception (e.g., the exception may change the running logic of the code) and the likelihood of the exception (e.g., malloc() is more likely to fail than mutex_lock).

I understand that as a student you worry about losing marks and want crystal clear instructions on which function should catch exceptions and which function should not. Let's face the reality that no such a guideline actually exists in your real life (i.e., your future job). As long as you use the common sense and catch some exceptions, you will not lose marks.

Below is a reasonable list:

Functions whose return values should be caught:

- pthread_cond_init
- pthread_cond_destroy
- pthread_mutex_init
- pthread_mutex_destroy
- pthread_create
- pthread_cancel

It is not mandatory to check the return values of the following calls:

- pthread_mutex_lock
- pthread_mutex_unlock
- pthread_cond_wait
- pthread_cond_signal
- pthread_cond_broadcast
- pthread_join

5. Unstable output: Without changine any code fragment, on the same input file, I run my acs twice yesterday and today. I found that their output is changing +-0.01s, is this common?

Answer: You should NOT expect the identical output! We are using machine time for simulation. Small variations in runtimes are common due to a number of factors such as branch prediction succeeding/failing or CPU usage by other applications.

6. Do I have to implement customers as threads since a customer does nothing?

Answer: You might have a different design that does not implement a customer as a thread. Nevertheless, implementing customers as threads makes the design logic simpler and cleaner. Actually, a customer is not doing nothing. It first needs to sleep until its arrival time. Refer to Q5, the sleep may cause minor difference in your final output.