ASP Activity 2

Contributors

1. Emmett Kogan

Q1 - D

Again there is no relation between t4 and t5 so either could be later. Main thread waits on both threads to finish before returning, and both threads just print and return. So the time should be something like main starts, pt1 starts, pt2 starts, both pt end, main ends D is valid A isn't valid because multiple things are happening at once I guess? I'm not certain if that is against the rules of the problem or not. For A B and C the main thread terminates at the same time of the other threads which can't be true.

Q2 - C

Main calls pthread join on thread 2 so at the very least main will exit after thread 2. pthread 2 calls join on thread 1 so the order of exits must be t1 then t2 then Main

There is no relation between t5 and t6 except that they are both between t4 and t7, so t6 could be less than t5 here for instance so C is the only one that is valid

Q3 -

a. If thread 2 enters the loop but is swapped out prior to picking up the lock, thread 1 can fully execute, so thread 2 would be blocked forever on the cond_wait when it reaches it
b. Again, there is a case where thread 2 simply never exits because it is forever blocked on the conditional variable when thread 1 reaches cond_signal prior to thread 2 reaching cond_wait.

There is also a data race on the done variable as thread 2 reads it outside of the critical section associated with it. But the worst case for this since done can't be de-asserted is that thread 2 loops again which *should* be harmless.

Q4 - B, E, F

Man page for pthread_exit states: `To allow other threads to continue execution, the main thread should terminate by calling pthread_exit() rather than exit(3).`

Q5 - C, D, F

A and B don't work for multiple producers/consumers. E introduces a data race on numItems since it accesses it outside the critical section, but, F does not have this problem since it is only cond_signaling outside of the critical section which would happen either way.