Part 6d. Producer/consumer implementation with locks and semaphores [20 points]

Assume you have the following code for accessing a shared-buffer that contains `max` elements (for some very large value of max). Assume multiple producer and multiple consumer threads access these routines concurrently. Assume the initial state is that the mutex is not held and that all buffers are empty. Assume the semaphore `empty` is initialized to 0 and `fill` is initialized to `max`.

```
void *producer(void *arg) {
            Mutex_lock(&m);                          // p1
            if (numfull == max)                      // p2
                  sema_wait(&empty);                 // p3
            do_fill(i);                              // p4
            sema_post(&fill);                        // p5
            Mutex_unlock(&m);                        // p6
}
void *consumer(void *arg) {
            Mutex_lock(&m);                          // c1
            if (numfull == 0)                        // c2
                  sema_wait(&fill);                  // c3
            int tmp = do_get();                      // c4
            sema_post(&empty);                       // c5
            Mutex_unlock(&m);                        // c6
 }
```

79) After the instruction stream "PPPPP" (i.e., after the scheduler runs 5 lines producer() for a producer thread P), which line of acquire will be executed for P when P is scheduled again?

a) P1
b) P3
c) P5
d) Some line after P6
e) None of the above

80) Assume the scheduler continues on with CCCCC (i.e., the scheduler runs 5 lines of consumer() for a consumer thread C and the full instruction stream is PPPPPCCCCC). Which line will thread C execute when it is scheduled again?

a) c1
b) c3
c) c5
d) Some line after c6
e) None of the above

81) Assume the scheduler starts another consumer with OOO (i.e., the full instruction stream is PPPPPCCCCCOOO). Which line will thread O execute when it is scheduled again?

a) c1
b) c3
c) c4
d) Some line after c6
e) None of the above

82) Assume the scheduler starts another producer with RRR (i.e., the full instruction stream is PPPPPCCCCCOOORRR). Which line will thread R execute when it is scheduled again?

a) p1
b) p3
c) p5
d) Some line after p6
e) None of the above

83) If a problem exists in the above code, what would be the easiest solution to fix it?

a) There is no problem with this code
b) Remove the calls to mutex_lock() and mutex_unlock()
c) Correct how the two semaphores were initialized
d) Change the semaphores to condition variables
e) None of the above