11. (14 pts) Multithreading
Consider the following attempt to implement producer/consumer pattern w/ Java 1.4.

```java
class Buffer {
    Buffer () {
        Object buf = null;
        boolean empty = true;
    }
    void produce(o) {
        synchronize (buf) {
            if (!empty) wait( );
            empty = false;
            notifyAll( );
            buf = o;
        }
    }
    Object consume( ) {
        synchronize (buf) {
            if (empty) wait( );
            empty = true;
            notifyAll( );
            return buf; // also releases lock
        }
    }
}
```

In the following, give schedules as a list of thread name/line number/range pairs, e.g., (t1, 1-4), (t2, 1), (t3, 5-8). For instance, one schedule under which x=1 and y=2 is (t1, 1-4), (t3, 5-8), (t2, 1-4), (t4, 5-8)

a. (2 pts) Give a schedule under which x = 2 and y = 1.
   (t2, 1-4), (t3,5-8), (t1,1-4), (t4,5-8)   etc…
   Have t3 execute after t2, and t4 execute after t1

b. (4 pts) Give a schedule under which x = 2 and y = 2, or argue that no such schedule is possible.
   (t4,5), (t2, 1-4), (t3,5-8), (t4,6-8) OR (t3,5), (t2, 1-4), (t4,5-8), (t3,6-8) etc…
   Have one consumer thread (either t3 or t4) misbehave by waiting on 5, then returning and continuing execution even though condition is not valid (i.e., empty = true), causing it to read value 2 already read by other consumer

c. (8 pts) Explain why the given Java code allows data races and why deadlock may occur.
   Because wait is not called in a while loop. Multiple threads may be woken, so the condition may not be true by the time a thread wakes up.

Deadlock:
(t1,1-4), (t2, 1), (t6, 1), (t3,5-8), (t2,2-4), (t6,2-4), (t5,5-8), (t4,5) etc…
Deadlock can occur by having producer thread t2 & t6 misbehave by waiting on 1, then returning and continuing execution even though condition is not valid (i.e., empty = false), for both, causing one producer to overwrite buf value already produced by other producer thread. One consumer will then hang because there are insufficient producers.