12. (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
        }
    }
}
t1 = Thread.run { produce(1); }
t2 = Thread.run { produce(2); }
t3 = Thread.run { x = consume( ); }
t4 = Thread.run { y = consume( ); }
```

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.

b. (4 pts) Give a schedule under which x = 2 and y = 2, or argue that no such
   schedule is possible.

c. (8 pts) Explain why the given Java code allows data races and why deadlock may
   occur.