1. (8 pts) Locks and synchronization
   a. (2 pts) When are condition variables needed in multithreaded programs?

   **When multiple threads need to wait for shared data in a producer/consumer relationship.**

   b. (3 pts) Why must condition variables be associated with a lock variable?

   **Because threads waking up (returning from wait) must reacquire the lock used to guard the shared data.**

   c. (3 pts) When would multiple condition variables be associated with the same lock variable?

   **When there are different groups of threads waiting for shared data, and only threads in a particular group (associated with the same conditional variable) should be woken up.**
2. (12 pts) Multithreading

Consider the following multithreaded Java 1.4 code:

```java
class Buffer {
    Buffer () {
        Object buf = null;
        boolean empty = true;
    }

    void produce(o) {
        synchronize (buf) {
            1. if (!empty) wait ( );
            2. empty = false;
            3. notifyAll ( );
            4. buf = o;
        }
    }

    Object consume( ) {
        synchronize (buf) {
            5. if (empty) wait ( );
            6. empty = true;
            7. notifyAll ( );
            8. return buf;
        }
    }
}

t1 = Thread.run { produce(1); }
t2 = Thread.run { produce(2); }
t3 = Thread.run { produce(3); }
t4 = Thread.run { x = consume( ); }
t5 = Thread.run { y = consume( ); }
```

Assume thread schedules are given as a list of thread name/line number/range pairs, e.g., (t1, 1), (t4, 5-8), would mean thread 1 executed line 1, followed by thread 4 executing lines 5-8.

For each of the following schedules, determine whether the schedule is possible. If it is, determine what values are assigned to x & y. If the schedule is not possible, explain why.

a. (3 pts)  (t1, 1-4), (t5, 5-8), (t3, 1-4), (t4, 5-8), (t2, 1-4)
    x = 3, y = 1
    Thread t1 puts 1 in buffer, thread t5 reads 1 from buffer, thread t3 puts 3 in buffer, thread t4 reads 3 from buffer, thread t2 puts 2 in buffer.

b. (3 pts)  (t1, 1-4), (t2, 1), (t3, 1), (t4, 5-8), (t5, 5), (t2, 2-4), (t5, 6-8), (t3, 2-4)
    x = 1, y = 2
    Thread t1 puts 1 in buffer, thread t2 waits since buffer is full, thread t3 waits since buffer is full, thread t4 reads 1 from buffer, thread t5 waits since buffer is empty, thread t2 wakes and puts 2 in buffer, thread t5 wakes and reads 2, thread t3 wakes and puts 3 in buffer.

c. (3 pts)  (t1, 1-4), (t2, 1), (t3, 1), (t4, 5-7), (t2, 2-4), (t4, 8), (t5, 5-8), (t3, 2-4)
    Not possible
    Thread t1 puts 1 in buffer, thread t2 waits since buffer is full, thread t3 waits since buffer is full, thread t4 reads 1 from buffer and calls notifyAll, thread t2 wakes and puts 2 in buffer [not possible since t4 has not released lock]...

d. (3 pts)  (t4, 5), (t5, 5), (t2, 1-4), (t5, 6-8), (t4, 6-8), (t1, 1-4), (t3, 1)
    x = 2, y = 2
    Thread t4 waits since buffer is empty, thread t5 waits since buffer is empty, thread t2 puts 2 in buffer, thread t5 waits and reads 2, thread t4 wakes and reads 2 (does not check whether buffer is empty since wait is not in a while loop), thread t1 puts 1 in buffer, thread t3 waits since buffer is full.