1. (10 pts) Lambda calculus
   (2 pts each) Use beta-reduction to fully evaluate the following λ-expressions.
   a. (λx.x) (λ.y.y) (λz.z)
      (λx.x) (λy.y) (λz.z) → (λy.y) (λz.z) → λz.z
   b. (λx.λy.x) y
      (λx.λy.x) y → (λx.λz.x) y → λz.y

   (Note: just the final answer for the above is also correct)

c. (6 pts) Show that the Church numeral encoding of 1*1 evaluates, after 0 or more beta reductions, to the encoding for 1. The encodings are in the box below. In your reduction sequence you may write ~ whenever you replace a definition from the box with its right-hand-side, or vice-versa. Write => when you perform a beta reduction. Here is a bit to get you started:

   M * N = λx.(M (N x))

   0 = λf.λy.y
   1 = λf.λy.f y
   2 = λf.λy.f (f y)
   3 = λf.λy.f (f (f y))

   1 * 1 ~
   λx.(1 (1 x)) ~ (by definition of *)
   λx.(1 ((λf.λy.f y) x)) => (by definition of 1)
   λx.(1 (λy.x y)) ~ (beta reduction)
   λx.(λf.λy.f y (λy.x y)) => (by definition of 1) 2 pts
   λx.(λy.(λy.x y) y) => (beta reduction) 2 pts
   λx.λy.x y ~ (beta reduction) 1 pt
   1 ~ (by definition of 1, and α-equivalence) 1 pt
2. (10 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) {
            while (!empty) wait( );
        }
        synchronize (buf) {
            empty = false;
            notifyAll( );
            buf = o;
        }
    }
    Object consume( ) {
        synchronize (buf) {
            while (empty) wait( );
        }
        synchronize (buf) {
            empty = true;
            notifyAll( );
            return buf; // also releases lock
        }
    }
}
```

(t1, 1-4), (t2, 1), (t3, 5-8) …

(We are abusing the thread creation notation here, and the call to produce() is not type correct—assume produce(1) is like produce(new Integer(1)).) 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) …

a. (2 pts) Give a schedule under which x = 1 and y = 2.
   (t1, 1-4), (t3, 5-8), (t2, 1-4), (t4, 5-8) OR
   (t2, 1-4), (t4, 5-8), (t1, 1-4), (t3, 5-8) OR …
   Key: (t1, 4) < (t3,8) & (t2, 4) < (t4,8)

b. (2 pts) Give a schedule under which x = 2 and y = 1.
   (t1, 1-4), (t4, 5-8), (t2, 1-4), (t3, 5-8) OR
   (t2, 1-4), (t3, 5-8), (t1, 1-4), (t4, 5-8) OR …
   Key: (t1, 4) < (t4,8) & (t2, 4) < (t3,8)

c. (3 pts) Give a schedule under which x = 1 and y = 1, or argue that no such schedule is possible.
   (t1, 1-4), (t3, 5), (t4, 5), (t3, 6-8), (t4, 6-8), (t2, 1-4) OR …
   Key: (t3, 5), (t4, 5) < (t3,6), (t4,6) -> both t3, t4 consume same item

d. (3 pts) Give a schedule under which x = 2 and thread 4 never terminates.
   (t1, 1), (t2, 1), (t1, 2-4), (t2, 2-4), (t3, 5-8), (t4, 5) OR …
   Key: (t1, 1), (t2,1) < (t1, 2), (t2,2) -> both t1, t2 produce at same time