Quiz 1 *Name:* ______________________

9/24/12

Examine the code below and tell us the result that will be printed by each of the four `println()`s found in `main()`. For full credit, outline your reasoning for each answer.

```
class Test {
    def f(x:Long):Long = x < 2 ? 1+x : f(x-1)*f(x-2);

    def g(m:Long):Long {
        var result:Long=2L, a:Long=1L;
        for (y in 2L..m) {
            val b=result;
            result *=a;
            a=b;
        }
        return result;
    }

    def p(m:Long):Long {
        var result:Long=2L, a:Long=1L;
        finish for (y in 2L..m) async {
                val b=result;
                atomic {
                    result *=a;
                    a=b;
                }
            }
        return result;
    }

    def a(m:Long):Long {
        val result=new Cell[Long](2L), a=new Cell[Long](1L);
        val P=Place.MAX_PLACES;
        finish for (y in 2L..m) at (Place((y%P) as Int)) async {
                val b=result();
                atomic {
                    result() *=a();
                    a()=b;
                }
            }
        return result();
    }

    public static def main(args:Rail[String]) {
        val x = new Test();
        val v= args.size>0? Long.parseLong(args(0)):5L;
        val ans=x.f(v);
        Console.OUT.println(ans);          // Line 0
        Console.OUT.println(ans==x.g(v));  // Line 1
        Console.OUT.println(ans==x.p(v));  // Line 2
        Console.OUT.println(ans==x.a(v));  // Line 3
    }
}
```

Line 0 = 32
$f(0) = 1$
$f(1) = 2$
$f(2) = 2 \cdot 1 = 2$
$f(3) = 2 \cdot 2 = 4$
$f(4) = 2 \cdot 4 = 8$
$f(5) = 8 \cdot 4 = 32$

Line 1 = true
This iterative serial code performs the same computation as the recursive serial code in `f()`.

Line 2 = ?
This parallel implementation of the iterative algorithm will compute the same as the serial when the loop bodies are executed atomically. However, this is not guaranteed as only part of the body is actually atomic. Therefore, this code is nondeterministic, with some interleavings yielding 32 and others not.

Line 3 = false
This multiplace implementation *will not* return the correct result. To begin with, the result from Place 0 is returned – results from other places are never shared back. Moreover, the values will be incorrect in many places as the code assumes global references to a and `result` which are not `GlobalRef`s. Finally the atomicity issue that was present in the previous part remains in this one.