

Name: _____

Section: _____

Quiz #7

1. A thread must hold a lock on an object before calling `wait` on it.

Circle one: TRUE FALSE

2. A thread must hold a lock on an object before calling `notifyAll` on it.

Circle one: TRUE FALSE

3. When `wait` is called on an object, the current thread releases the lock on the object.

Circle one: TRUE FALSE

4. When `notifyAll` is called on an object, the current thread releases the lock on the object.

Circle one: TRUE FALSE

5. When a thread holds a lock on an object, it is impossible for any other threads to modify that object.

Circle one: TRUE FALSE

6. When a thread holds a lock on an object, that thread cannot be interrupted until it releases the lock.

Circle one: TRUE FALSE

7. It is possible for one thread to hold locks on two different objects at the same time.

Circle one: TRUE FALSE

8. It is possible for two different threads to simultaneously hold a lock on the same object.

Circle one: TRUE FALSE

[More on the back page...]

9. (15 points) Decide whether each of the following methods could cause problems when the `foo` method is executed by multiple threads concurrently. (Recall that a “data race” is a situation in which multiple threads attempt conflicting updates on a shared resource.)

<pre>static int a; static void foo() { a++; }</pre>	Could this code cause data races ? YES / NO
<pre>static int a; static final Object lock = new Object(); static void foo() { synchronized(lock) { a++; } }</pre>	Could this code cause data races ? YES / NO
<pre>static int a; static void foo() { final Object lock = new Object(); synchronized(lock) { a++; } }</pre>	Could this code cause data races ? YES / NO
<pre>static final Object a = new Object(); static final Object b = new Object(); static void foo() { synchronized(a) { synchronized(b) { } } }</pre>	Could this code cause deadlock ? YES / NO
<pre>static void foo(Object a, Object b) { synchronized(a) { synchronized(b) { } } }</pre>	Could this code cause deadlock ? YES / NO