Define “deterministic”.

Is a program that uses a random number generator like `java.lang.Math.random()` deterministic or non-deterministic? Deterministic. Why?

Define “monitor”.

What does this phrase in paragraph 3 mean “executing its body”?

What is “spinning”?

What is “busy waiting”?

Does spinning ever a good approach? Why?

Draw a diagram that illustrates the Terminology and Framework in 17.1.

Draw a diagram that illustrates the transitivity of the relationship “precede”.

What does "B must intervene between A and C" mean? Illustrate it in a precedence diagram.

Take the diagram of the Terminology and Framework and use it to illustrate a couple of the rules in the first bullet list in Section 17.3 by labeling the actions and indicating their precedence relationships. Also illustrate them in space-time diagrams.

What does this code print?

```java
class Test {
    public static void main(String[] args) {
        Test t = new Test();
        synchronized(t) {
            synchronized(t) {
                System.out.println("made it!");
            }
        }
    }
}
```

Hint: see the first bullet in 17.5.

What does this mean: “With respect to a lock, the lock and unlock actions performed by all threads are performed in some total sequential order. This total order must be consistent with the total order on the actions of each thread.”

Why are there volatile variables?

---

1 JLS, Section 14.18.

2 JLS, Section 17.5.
In Section 17.9 the JLS says “Locking any lock conceptually flushes all variables from a thread's working memory, and unlocking any lock forces the writing out to main memory of all variables that the thread has assigned.” This sounds expensive if threads have many variables and frequently lock or unlock. How should a programmer try to mitigate this cost?

Reading my lock.java

Why did I bother to write lock.java? Why wouldn't a programmer just use the lock associated with each Java object, or synchronized statements?

The comment on release_read_lock says “Assumes the thread has a read lock”. Suppose that instead of making this assumption the method wanted to check the assumption and throw an exception if it were not true. How could this be done?

What's the main problem with the class testLockCode?