CS162 Section 2 (9/13/12)

True/False

1. A thread needs to own a semaphore, meaning the thread has called semaphore.P(), before it can call semaphore.V()

2. A thread needs to own the monitor lock before it can signal() a condition variable.

3. Anything that can be done with monitors can also be done with semaphores.

Short Answers

1. Give two reasons why the following implementation of a condition variable is incorrect (assume that MySemi is a semaphore initialized to 0):

```
Wait()   { MySemi.P(); }
Signal() { MySemi.V(); }
```

2. What is the difference between Mesa and Hoare scheduling for monitors?

3. With spinlocks, threads spin in a loop (busy waiting) until the lock is freed. In class we argued that spinlocks were a bad idea because they can waste a lot of processor cycles. The alternative is to put a waiting process to sleep while it is waiting for the lock (using a blocking lock). Contrary to what we implied in class, there are cases in which spinlocks would be more efficient than blocking locks. Give a circumstance in which this is true and explain why a spinlock is more efficient.

4. Give two reasons why this is a bad implementation for a lock:

```
lock.acquire() { disable interrupts; }
lock.release() { enable interrupts; }
```

Synchronization

Show how to implement the Semaphore class using Monitors (i.e. the Lock and CondVar classes). Make sure to implement both P() and V(). None of the methods should require more than five lines. Assume that Monitors are Mesa scheduled.

```
public class Semaphore {
   Lock lock;   // Every Monitor has a lock and CondVar
   CondVar c;
   Int value;   // Semaphores have an integer value

   public Semaphore(int initialValue) {
      value = initialValue;
      lock = new Lock();
      c = new CondVar(lock);
   }

   // P() and V() are distinct methods.  I just formatted them this way to save space.

   public P() {                                    public V() {
      // Your code here.                             // Your code here.

   }                                               }

}
```