# **CS 162 Section 3**

## 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. Changing the order of semaphores' operations in a program does not matter.

#### Short Answer

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. Explain why semaphores can be used to do anything monitors are used for. If this is not true, explain why not.

## **Longer Answer**

### Semaphores

Implement the P() and V() methods of a Semaphore class backed by monitors (i.e. the Lock and CondVar classes). Neither 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);
}

public P() {
    public V() {
```

#### **Producer and Consumer**

Consider the following two functions implementing a producer and consumer by using monitors:

```
void send(item) {
    lock.acquire();
    enqueue(item);
    printf("before signal()\n");
    dataready.signal(&lock);
    printf("after signal()\n");
    lock.release();
}
item = get() {
    lock.acquire();
    while (queue.isEmpty()) {
        printf("before wait()\n");
        dataready.wait(&lock);
       printf("after wait()\n");
    item = dequeue();
    lock.release();
}
a. Assume two threads T1 and T2, as follows:
                    item = get();
   send(item);
```

What are the possible outputs if the monitor uses the Hoare implementation?

- b. Repeat question (a) for a Mesa implementation of the monitor.

What are the possible outputs if the monitor uses the Hoare implementation? Please specify from which thread does an output come by specifying the thread id in front of the output line, e.g., [T1] before signal or [T2] after wait.