Suggested answers to Week 4 assignments

Activity 4.1, 4.2

Activity 4.1

1. Could use hardware getAndSet instruction, software Peterson's Algorithm, etc, to provide atomicity for acquire() and release().
2. The operations acquire() and release() (the non-busy wait versions on Slide 6.24) are short, so a busy waiting solution for them is acceptable (the busy wait won't take long and so won't waste many CPU cycles).

Activity 4.2

- **P1**. Ensure mutual exclusion in updating the shared buffer. **P2**. For producer, ensure there's at least one empty buffer slot. **P3**. For consumer, ensure there's at least one full buffer slot.
- Three semaphores, one for each of P1, P2, P3.
  - One binary semaphore for the *mutual exclusion* problem P1. Two counting semaphores for the two *condition synchronization* problems P2 & P3.
- Call the semaphores mutex, empty, full for P1, P2, P3, respectively. Initialize them to 1, N, 0.

Activity 4.2 (cont'd)

```
public void produce(Work item) {

        acquire(empty);
        acquire(mutex);

        buffer[in] = item;
        in = (in + 1) % BUFFER_SIZE;

        release(mutex);
        release(full);
    }
```

Code for consumer is exactly analogous (exchange the places of empty and full).