CMPS 111 Operating Systems
Winter 2018, Homework #3

Aaron Steele, atsteele@ucsc.edu

Question 1

Deadlock: A situation from which it is impossible to proceed.
The classic example is the Dining Philosopher's Problem. There are 5 philosophers at a table. There are 5 chopsticks. A philosopher can only eat or think, and they can only eat when they have 2 chopsticks. If they all immediately pick up all the chopsticks they can, they will probably all pick up one, and then they will wait for another philosopher to drop their chopsticks so they can pick it up, except nobody ever drops their chopsticks. So, a deadlock.

The conditions required for a deadlock are at least two processes and at least two resources. The first process has a resource the second needs to continue, and the second has a resource the first needs to continue. Thus, there is a circular dependency between the threads, which creates a deadlock.

Question 2

The pseudocode has no mention that the two resources are required to complete the modifications to said resources, so something like this should produce the same result:

```
Process1 {
        acquire(lockA);
        modify(resourceA);
        release(lockA);
        acquire(lockB);
        modify(resourceB);
        release(lockB);
}
Process2 {
        acquire(lockA);
        modify(resourceA);
        release(lockA);
        acquire(lockB);
```

```
        modify(resourceB);
        release(lockB);
}
```

My reasoning here is that in the modify() command, there is no mention of using resource A in modify(resourceB), for example. This means the operations are independent of each other, and so the acquire() statements can be separated, meaning each thread only needs one resource at once. No circular dependency can be created with a single resource, which means that any possible deadlock in this example has been resolved.

Question 3

Question 4

Question 5