Jeremy Neal
CMSC 421 - Park
Homework 2
1 April 2015

Question 2:
Part A:

```
int lock(int *val) {
    int success = TestAndSet(&val);
    if (success == 0)
        return 1;
    return 0;
}

void unlock(int *val) {
    int success = TestAndSet(&val);
    if (success == 1)
        return 1;
    return 0;
}
```

Part B:
Assuming that TestAndSet(int *val) works atomically, this will ensure mutual exclusion. This particular implementation assumes some behavior about TestAndSet: there are two states (0: locked, 1: unlocked). Each time the function is called, it returns the new value of the val variable passed to the function. This value is then checked inside each function to ensure the correct value has been returned.

Question 3:
The issue here is due to the fact that increment and decrement operations are not atomic. The first check the value, mutate it it, then return it. As such, since x isn't locked when it is either being incremented or decremented, we can't know what the value will necessarily be. This is called a race condition.

Let's assume that incrementing occurs first. First, x will be checked (0), it will then be incremented (1), and returned. However, we don't know what value the decrement operation sees x as. If it checks it before x is stored, it will read 0 and return -1.

Under the best circumstances, x will be 0 at the end. However, the value could vary within the range of -1 or +1, as at most, there could be one out of order read that could affect the value incorrectly.