## Operating Systems

## Test 1 - Part 2

Name: \_\_\_\_\_\_\_Nhan Tran\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_11/30/12\_\_\_\_\_\_\_\_

A description...

**Synchronization problem:** We will use locks to direct traffic. Cars come to the intersection, stop at the stop sign, and either continue in the direction they are driving, make a left turn, or make a right turn. Each quadrant of the intersection has a lock associated with it (NW, SW, NE, and SE). See Figure above.

**Correctness constraints:**

● No two cars should ever be heading in directions that cross. Keep in mind that a car making a left turn will be moving across the intersection (e.g., if the car in the above figure turns left, it will pass through the SW, SE, and NE quadrants of the intersection).

● A car’s path should also never intersect with another car’s path (i.e., a car moving EAST-to-SOUTH intersecting with a car travelling SOUTH-to-NORTH or NORTH-to-SOUTH).

You have been provided with **incomplete** code (intersection.c, intersection.h) that uses a pthreads lock for each quadrant and prints out the actions taken by a car. Before the action is taken (i.e., printed), *you must first acquire the necessary locks*. Read the man page documentation on POSIX pthreads locks and add the necessary code to correctly manage the locks so that there is correct operation and no deadlock. If the code exits successfully every time, the locking strategy was successful.