CIS 640 Spring 2009: Homework 1, Due February 9

1. Exercise 11 from Section 2.10 (page 41 of textbook)

2. Exercise 14 from Section 2.10 (page 42 of textbook)

3. Another way to generalize the two-thread Peterson lock to n threads is to arrange a number of two-thread Peterson locks in a binary tree. Suppose n is a power of two. Each thread is assigned a leaf lock which it shares with one other thread. Each lock treats one thread as thread 0 and the other as thread 1. In this tree-lock, the thread acquires every two-thread Peterson lock from that thread's leaf to the root. The tree-lock's unlock method unlocks each of the two-thread Peterson locks that thread has acquired, from the root back to its leaf.

   (a) Write a Java implementation of tree-lock that satisfies mutual exclusion and deadlock-freedom.

   (b) Benchmark your implementation with $n = 8$ and $n = 32$ threads and compare its performance to that of the filter lock. For each benchmark, construct an implementation of a critical section that protects a counter that is incremented 1, 10, and 100 times. Run each test 3 times. Report the average time it takes each of the two algorithms, the counter based on the tree and filter, to complete each of the counter increment tests.

   The course page has sample Java code for Peterson and filter locks. You should run the experiments on the multi-processor machine *arachnid*. Students officially registered for 640 should have a login to this machine (just use your standard SEAS login/password), otherwise request manager for an account mentioning 640 coursework. Submit the code as well as experimental results.