CIS 640 Spring 2009: Homework 2, Due March 4

1. Exercise 32 from Section 3.11

2. Exercise 124 from Section 10.9

3. We have n threads, each of which executes method `foo()` followed by `bar()`. We want to add synchronization to ensure that no thread starts executing `bar()` until all threads have finished executing `foo()`. To achieve this, we will insert some barrier code between the two methods. Implement the following two schemes for barrier code.

   (a) Use a shared counter protected by test-and-test-and-set lock. Each thread locks the counter, increments it, releases the lock, and repeatedly reads the counter until it reaches n.

   (b) Use an n-element Boolean array A. Initially all entries are 0. When thread 0 executes its barrier, it sets $b[0]$ to 1, and repeatedly reads $b[n-1]$ until it becomes 1. Every other thread i, repeatedly reads $b[i-1]$ until it becomes 1, then it sets $b[i]$ to 1, and repeatedly reads $b[n-1]$ until it becomes 1.

   Implement both these schemes in Java. Each of the methods `foo()` and `bar()` just sleeps for 20 milliseconds. Test the two schemes for $n = 16$. Run each scheme at least ten times, measure the total runtime in each test run, discard the highest and lowest values, take the average, and use it to compare the two schemes. Which performs better? Can you explain the reason?

   You should debug your code on eniac and then run the experiments on the multi-processor machine *acggrid34.seas.upenn.edu*. Submit the code as well as experimental results.