CS1760: Multiprocessor Synchronization Due 13 November 2023

Homework 5

Professor: Maurice Herlihy *TA: cs1760tas@lists.brown.edu*

Problem 1. Explain why the fine-grained locking algorithm is not subject to deadlock.

Problem 2. Using locks and conditions, implement a symmetric locking protocol for two types of threads: RED and BLUE. Assume you can query a thread's color by calling, for example,

```
1 if (Thread.getColor() == RED) { ... }
```

For correctness, never allow a RED and BLUE thread to enter simultaneously, but do allow multiple RED or multiple BLUE threads to hold the lock at the same time. For fairness, if a RED thread tries to acquire a lock currently held by BLUE threads, do not let any more BLUE threads in until the current BLUE threads have drained out and the RED thread has acquired the lock (and vice-versa for RED, of course).

Hint: use the ReadWriteLock in the textbook as a model.

Problem 3. A *savings account* object holds a non-negative balance, and provides deposit(k) and withdraw(k) methods, where deposit(k) adds k to the balance, and withdraw(k) subtracts k, if the balance is at least k, and otherwise blocks until the balance becomes k or greater.

Problem 4. Using locks and conditions, modify your *savings account* withdraw() method to support two kinds of withdrawals: *ordinary* and *preferred* such that no ordinary withdrawal occurs if there is a preferred withdrawal waiting to occur. (Assume withdraw() takes a Boolean preferred argument.)