Homework Exercise #2
Due: September 29, 2011

2. In some shared-memory algorithms, some shared data structures are protected by locks. Prior to accessing the shared data structure, a process must *acquire* the lock. The process *releases* the lock after accessing the shared data structure. Between acquiring the lock and releasing it, the process is said to *hold* the lock.

Here, we consider a single-writer multi-reader lock. Such a lock can be acquired by a process in one of two modes: *reading mode* or *writing mode*. (A process is supposed to acquire the lock in reading mode if it just wants to read information from the shared data structure, and the process is supposed to acquire the lock in writing mode if it wants to modify the shared data structure.) The lock should ensure the following safety property: at any time, if some process is holding the lock in writing mode, then no other process is holding the lock (in either mode). This is used to guarantee that two updates to the data structure do not interfere with each other, and it also ensures that a process cannot read inconsistent information from the data structure while another process is in the middle of updating it. Note that it *is* possible for multiple processes to hold the lock simultaneously in reading mode, as long as there is no process holding the lock in writing mode.

(a) Describe a reasonable progress condition that a lock should satisfy in order to be useful.

(b) Give a formal definition of an I/O automaton for a single-writer multi-reader lock. You need only model the lock, not the data structure that it protects. If you need to make any (reasonable) assumptions about the behaviour of the lock in order to model it, include the assumptions in your submission.