Homework 5:
More Concurrency

Homework in class on the date indicated above. Late homeworks will be penalized by $5 \cdot 2^{n-1} \%$, where $n$ is the number of days the assignment is late. Please indicate which problems, if any, took extra time.

Suggested reading: C. A. R. Hoare, Communicating Sequential Processes. CACM 21(8), August 1978. You can get it from http://citeseerx.ist.psu.edu/. (Careful: there is a book by the same title available. It is good and worth reading, but it is 260 pages. The paper is less than 12 pages.)

As in lecture, this assignment will use $[G \rightarrow S]$ to stand for $if \ G \rightarrow S \ fi$ as well as $*[G \rightarrow S]$ for $do \ G \rightarrow S \ od$ and also the abbreviations $[G]$ to mean $[G \rightarrow skip]$ and $*[S]$ to mean $*[true \rightarrow S]$.

1. Sequential review 20%

Prove termination of the following program. Initially, $x$ is any positive integer.

$*[\ odd(x) \land x \geq 3 \rightarrow x := x + 1$
$\sqcap even(x) \land x \geq 2 \rightarrow x := x / 2]$

2. The pebble game 20%

Consider a game involving one pebble on a 5×5 grid, $x \in [0, 4], y \in [0, 4]$. There are four concurrent players executing the programs

$L \equiv *[moveleft ]$
$R \equiv *[moveright ]$
$U \equiv *[moveup ]$
$D \equiv *[movedown ]$

Where moveleft is the fragment $x := x - 1$, moveup is $y := y + 1$, etc.

Introduce semaphores with specified initial values such that all actions of

$L \ || \ R \ || \ U \ || \ D$

are safely synchronized and such that the pebble does not fall off the edge of the grid. You should not need to modify the program beyond the introduction of $P$ and $V$ operations on a small number of semaphores. Prove that your program satisfies safety and progress conditions.

3. Fair mutual exclusion with unfair P and V operations 60%

As discussed in lecture, the simplest definitions of $P$ and $V$ give rise to what we call “one-sided” semaphores, for which the axioms are as follows.

A1. $s = s_0 + cV(s) - cP(s)$
A2. $s \geq 0$
A3. $s = 0 \lor qP(s) = 0$
A4. $qV(s) = 0$

Note that the axioms do not restrict which of several processes suspended on $P(s)$ is unblocked by a $V(s)$. Semaphores of this kind are called unfair since the implementation is allowed to pick the same
process every time it unblocks a process. A fair semaphore, on the other hand, would guarantee an upper bound on the number of times $V(s)$ can be executed during the time a process is blocked on $P(s)$.

i. Given only unfair $P$ and $V$ operations, show how to implement a fair semaphore. Assume that an unknown but bounded number $N$ processes may be accessing the semaphore. Your implementation should use only a fixed number of semaphores; this number should not scale with $N$. You will probably need several semaphores and a few integer variables.

ii. Modify your solution in such a way that the largest integers it has to store scale no faster than $N$ (that is, they do not keep counting higher and higher for every time your semaphore is accessed). Show that your solution is correct: that is, show how to derive the semaphore axioms above for your solution from the semaphore axioms for the unfair semaphores you use in the implementation; also prove that your implementation is fair, as required.

**Hints.** This may be a challenging problem. Study Dijkstra’s solution to the Dining Philosophers problem we discussed in class. Also keep in mind that the axiomatic definition of the $P$ and $V$ operations is quite strict. For instance, because we have defined $qP(s)$ so strictly, the following code for allowing another process to run and then “trying something again” is reasonable since if any other process is blocked on $s$ when $V(s)$ is executed, that process is unblocked before this process can execute $P(s)$:

$$[	ext{not my turn} ightarrow V(s); P(s)]$$