York University CSE6117 September 15, 2014

Homework Exercise #2
Due: 11:30 a.m., September 24, 2014

1. We looked at Dijkstra's solution to the N-process mutual exclusion problem in class. We assume the processes are numbered $1..N$. Below, it is given using more modern computer language constructs. (▷ indicates a comment.)

Shared variables:

$Interested[1..N]$	▷ Boolean entries indicate which processes are interested in using the resource
$Passed[1..N]$	▷ Boolean entries indicate which processes have passed Phase 1
K	▷ integer is the name of a process allowed to try for the resource next

Local variable:

done	▷ Boolean variable that is used to determine when process can exit outer loop

Code for process i:

```
▷ ENTRY SECTION
$Interested[i] \leftarrow$ true
loop
    loop                                  ▷ Entry Phase 1
        exit when $K = i$
        if not $Interested[K]$ then       ▷ try to go next
            $K \leftarrow i$
    end loop
    $Passed[i] \leftarrow$ true           ▷ begin Entry Phase 2
    $done \leftarrow$ true
    for $j \leftarrow 1..N$ except $i$    ▷ check if anyone else has passed Phase 1
        if $Passed[j]$ then               ▷ go back to Phase 1
            $Passed[i] \leftarrow$ false
            $done \leftarrow$ false
    end for
    exit when done
end loop

▷ CRITICAL SECTION

▷ EXIT SECTION
$Passed[i] \leftarrow$ false
$Interested[i] \leftarrow$ false

▷ REMAINDER SECTION
```

For the critical section, we assume that any process that enters the critical section (and does not experience a stopping failure) eventually completes the critical section and begins executing its exit section.

(a) The original paper lists many properties that the algorithm is supposed to satisfy, but does not really explain why all of them hold. For example, property (c) says "If any of the computers is stopped well outside its critical section, this is not allowed to lead to potential blocking of the others" and this point is not explicitly addressed in the proof of correctness. Let's examine what "well outside" means for this algorithm. The system is said to be *blocked* in an infinite execution if there is some suffix of the execution where some process performs infinitely many steps of the entry section and no process ever enters the critical section. Give a precise description of the conditions when a stopping failure of a process could cause the system to block. (For example, a possible answer might be that blocking could occur if and only if a process stops when it is just about to execute line 5 or line 10, and there is a full moon.)

Briefly explain why your answer is correct. That is, explain how stopping under the conditions you give could cause the system to block, and explain why stopping in any circumstance that does not satisfy your conditions cannot cause the system to block.

(b) The liveness property described in Dijkstra's paper is a bit weak. Here is a stronger liveness property:

For every execution α where no process stops taking steps and for every process p, if p is in its entry section at some time during α, then there is some later time during α where p enters its critical section.

Does Dijkstra's algorithm satisfy this stronger property? Prove your answer is correct.