CSE5XX Distributed Computing

2020 Winter • EndSem • 100 points

Given to you: May 24, 2020, 8:00 PM;
Due: May 25, 11:59 PM.

This take-home exam permits the use of a Linux/Windows laptop/desktop running javac, g++, Eclipse, Idea, and other software development and drawing tools. It is otherwise a traditional closed book, closed notes exam. In particular, you are honor bound **not** to Internet surf or access any content already existing (other than as indicated) once you access the exam until you turnin the answers. The primary reasons in making this a take-home are to relieve time pressure and give you a comfortable (computing/home) environment. Do not give or take help from others. Submit your `answers.pdf` in the Google Classroom. *Not* as an email attachment.

I. (5 points each) The following statements may or may not be (fully or partially) valid. Explain the underlined technical terms occurring in each statement. Explain/ discuss/ dispute the statement. It is *possible* to write no more than, say, five, lines each, and yet receive full score.

1. A "safety" property is defined thus: Let **bad** be a predicate characterizing a "bad" state of program code segment S. Assume that {P} S {Q} holds. Assume that I is a global invariant. If I implies the negation of **bad**, we say that **not bad** is a safety property for S. Thus, it all depends on whether *we consider a certain property good or bad.*
2. In a collection of semaphores that constitutes a split binary semaphore, at least one of them must be 1.
3. The happened before relation (as in the Andrews book, e.g.) requires a logical clock for every process. But a typical machine (node) in a distributed system runs several processes. So, it is sufficient to maintain a logical clock per node that is shared by all processes running on that node.
4. "A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable." Is this a good definition?

II. (20 points each)

1. In distributed computing, we wish to maximize (i) concurrency, we prefer (ii) symmetric solutions, we rate (iii) correctness much higher than efficiency, and cannot tolerate (iv) deadlocks or (v) livelocks. (a) (10 points) Define and explain the five underlined technical terms. (b) (10 points) Defend *why* for each of these five expectations.
2. Consider the following await-statement based model of the classic Dining Philosophers problem. The index expression `i+1` below is modulo 5. The `up` and `dn` (down) refer to whether the philosopher picked the fork up or put the fork down. There are no fat- or parallel bars in this code. As usual, angular brackets enclose atomic operations.

```
var up[0..4], dn[0..4] initially all 0;
philosopher[i: 0..4]::
  do true →
     think;
     < await up[i] == dn[i] → up[i]++ > ;
     < await up[i+1] == dn[i+1] → up[i+1]++ > ;
     eat;
     < dn[i]++ > ;
     < dn[i+1]++ > ;
  od
```

(a)(10 points) Using the Pass-the-Baton technique systematically convert it to an equivalent algorithm using semaphores only and no awaits. (b)(10 points) Is this code free from starvation? Disregard if the above is a "solution" or not [in this question]. Explain your answers fully.

3. Do any of the temporal assertions below for the code block at left hold? Where? When? How? Under weakly fair scheduling? Under strongly fair? Note the fat-bar []. Mathematical reasoning is expected. Intuitive reasoning will deprive you of full score.

```
int x := 0; int y := 1;
do true → x := x + 2;
[] true → y := y + 3;
od
```

(a) (5 points) always x % 2 == 0
(b) (5 points) eventually x > 55
(c) (5 points) eventually always x > 44
(d) (5 points) sometimes y % 2 == 0

III. (0 points) [For survey purposes only.] Please record your effort in minutes for each of the above eight items. Other feedback you wish to give is also welcome.
