# CEG 7370 Distributed Computing

## 2014 Spring • MidSem • 100 points

Given to you: Mar 20, around 9:30 PM;
Due: Mar 21, 11:59 PM.

This take-home exam permits the use of a Linux/Windows laptop/PC 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 final 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 on `thor.cs.wright.edu` using `~pmateti/ceg7370/turnin MT answers.pdf`

1. (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.
   i. The semantics of the fat-bar operator [] permits us to equivalently replace S1 [] S2 with simply S1.
   ii. ``Every philosopher will eventually get hungry.'' is a safety, not liveness, property.
   iii. *Asynchronous* message passing is more fundamental than *synchronous* message passing.
   iv. Consider the program segment given here.

      Determine *P* a predicate that characterizes the weakest deadlock-free precondition for the program. Also, explain how you arrived at P.

      ```
      co <await x >= 7 -> x := x - 3>
      || <await x >= 6 -> x := x + 1>
      || <await x >= 5 -> x := x + 1>
      oc
      ```

   v. Compute *wp(S, i = 5)*, where S is

      ```
      if i < 5 --> i := i + 1 [] i > 5 --> i := 5 fi
      ```

2. (15 points each)
   i. In the distributed programming context, `P(m); Critical-Section; V(m)` is not starvation-free. Give a detailed starvation scenario. What does Morris' algorithm to eliminate this scenario?
   ii. Construct the ssend(pid, msg) and srecv(pid, msg) primitives of synchronous message passing from asend(pid, msg) and arecv(pid, msg) primitives of asynchronous message passing.
   iii. Reconstruct the binary "tree" serialized below. List each node as a triplet of (info integer, left-link and right-link), as usual. The first digit in the sequence is the offset of the root node.

      ```
      marshalled sequence: 2 3 4 7 4 7 8 4 A 1 4 2 7
      offsets in hex:      1 2 3 4 5 6 7 8 9 A B C D
      ```

   iv. The *Recursive Data Representation: Small Set of Integers* of Hoare's CSP paper is reproduced below. Extend it to respond to a command `S(1)!remove(x)` from S(0) that removes x from the set, if it contains it, and does nothing if it does not.

      ```
      S(i: 1 .. 100) ::
      *[   n: integer; S(i-1)?has(n) --> S(0)!false
       []  n: integer; S(i-1)?insert(n) -->
      ```

```
*[  m: integer; S(i-1)?has(m) -->
        [   m <=  n --> S(0)!(m = n)
        []  m >   n --> S(i+1)!has(m)
        ]
 [] m: integer; S(i-1)?insert(m) -->
        [  m < n --> S(i+1)!insert(n); n := m
        [] m = n -->  skip
        [] m > n --> S(i+1)!insert(m)
] ] ]
```

v. In a bag *B* of *b* integers, it is known that each element appears exactly *4* times. We wish to delete the duplicates resulting in a set of only *b/4* distinct integers that occurred in the bag. Design, and implement a solution in C/Java-Linda. Assume that the tuple space already contains `<"B", x_i>` for all $x_i$ in B, and the size of the bag `<"b", b>`. You lose 5 points for each use of `inp` or `rdp`. You are expected to present your design and implementation *with full explanations*. As usual, we wish to maximize concurrency, we prefer a symmetric solution, and we must have correctness, we must not have deadlocks or livelocks.

3. (0 points) [For survey purposes only.] Please record your effort in minutes for each of the above ten items. Other feedback you wish to give is also welcome.

---