CpSc 418 **Midterm** October 18, 2012

100 points

Time for the exam: 80 minutes.

Open book: anything printed on paper may be brought to the exam and used during the exam. This includes the textbook, other books, printed copies of the lecture slides, lecture notes, homework and solutions, and any other material that a student chooses to bring.

Calculators are allowed: no restriction on programmability or graphing. There are a few simple calculations needed in the exam, a calculator will be handy, but the fancy features will not make a difference.

No communication devices: That's right. You may not use your cell phone for voice, text, web-surfing, or any other purpose. Likewise, the use of computers, iPods, etc. is not permitted during the exam.

Good luck!

CpSc 418 **Midterm** October 18, 2012

100 points

Answer all **five** questions.

1. **Erlang (15 points + 3 extra credit).**
   Draw a line joining each Erlang expression on the left to the value obtained by evaluating that expression on the right.
   Grading: 3 points for each correct line, -1 point for each incorrect line.

Expression	Value
`lists:map(fun(X) -> 2*X end, [1, 2, 3])`	
	`3`

`lists:foldl(`
`  fun(A, B) -> A*B end,`
`  1, [2, 3, 4]`
`)`
	`5`

`(fun(N) ->`
`  G = fun(H, M) ->`
`  case M of`
`  0 -> 0;`
`  _ -> M+H(H,M-1)`
`  end`
`  end,`
`  G(G, N)`
`end)(4)`
	`10`
	`24`

`2 + 3`	`32`

`lists:foldl(`
`  fun({X, Y}, A) -> A + X*Y end, 0,`
`  lists:zip([1, 2, 3], [4, 5, 6])`
`)`
	`[2, 4, 6]`

`length([1,2,3])`	

2. **DLS Talk (10 points).**
What is a Craig Interpolant (check one):

- ☐ Craig Interpolants are a method for optimizing functional programs. The compiler computes the interval between when a variable is declared and when it is last read. If a new "copy" is made following the last read in order to modify a field, the compiler eliminates the copy and just modifies the existing value.
- ☐ If P and Q are boolean-valued formulas, and there is no way to satisfy both P and Q, then a Craig Interpolant is a boolean-valued formula, I, such that: $P \Rightarrow I$; $I \Rightarrow \neg Q$; and all variables that appear in I appear *both* P and Q.
- ☐ Given a list of points (x_1, y_1), (x_2, y_2), $\ldots (x_n, y_n)$ with $x_1 < x_2 < \cdots < x_n$, a Craig Interpolant is sequence of polynomials, P_1, P_2, $\ldots$, P_{N-1} for interpolating between values of the data. They are similar to cubic-splines but guarantee that interpolated values are bounded above and below by the values of the original data.
- ☐ Given an old API for Windows, or Linux, or Java, or Erlang, etc., and given a new version of the API, a Craig Interpolant is a structured way of writing a interpretation layer that allows programs that were written for the old API to run on a system with the new version.

3. **Synchronization (25 points)**

A mutual exclusion algorithm for two threads is *safe* if the two threads cannot be in their critical regions at the same time. An algorithm is *live* if any thread that tries to enter the critical region eventually does so.

Dekker's algorithm was designed to work with a shared memory where no processor is guaranteed uninterrupted access to the memory for a sequence of two or more reads or writes. For example, if a processor reads a memory location and then writes a new value to that location, other processors may access the same memory location between the read and the write. We say that a memory location is shared if it can be read and/or written by both threads.

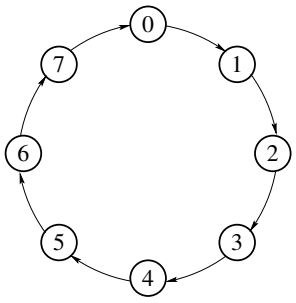
In this problem, you will show that any mutual exclusion algorithm that is safe and live must use at least two shared memory locations. You will do this by showing that an algorithm that uses only one shared memory location cannot guarantee safety.

(a) **(8 points)** Consider the part of execution between when a thread leaves its non-critical code and when it enters its critical section. I will say that a thread in such a state is "trying to enter." Show that each thread must read the shared location at least once when trying to enter.

(b) **(5 points)** Show that each thread must write the shared location at least once when trying to enter.

(c) **(12 points)** Consider a scenario when both threads try to enter at the same time. Note that each thread must have its first write of the shared location and its last read. Using this (or you can come up with your own clever approach), describe an ordering of events that leads to a violation of mutual exclusion (assuming liveness).

Hint: The liveness requirement is only needed in the proof above to rule-out some rather silly "implementations." For example, a algorithm that never lets either thread enter its critical region is safe but not live.



4. **Message Passing (25 points)**

   (a) (10 points) Consider the simple ring network shown in the figure above. For simplicity, messages only travel clock-wise in the ring. Assume that each link can transfer a message in one unit of time. Now consider the program:

```
0. parallel_for(int i = 0; i < 8; i++) {
1.   while(true) {
2.     for(int j = 0; j < 8; j++) {
3.       if(i ≠ j) {
4.         node i sends a message to node j;
5.       }
6.     }
7.     node i receives any messages that have arrived.
8.   }
9. }
```

   In other words, each node repeatedly sends messages to all of the other nodes. The receive at line 7 is just to make sure that all messages are delivered so that the network is never blocked waiting for a node to accept a message. The nodes have a enough buffering that they can be constantly offering traffic to the network. Assume that the computations are performed arbitrarily fast; so, the performace is determined only by how fast the network can deliver messages.

   Show that in the long run, each of the eight nodes completes the while loop that spans lines 1–8 once every 28 time units.

(b) (10 points) Now, assume that each node takes time t_0 total to send one message and receive one message (regardless of the message length). Assume that a link can transfer a message of length w in w time units. Derive a formula per iteration of the while loop as a function of t_0, w.

(c) (5 points) Make a rough plot of the value of your formula for the case that $t_0 = 20$, and $0 \leq w \leq 20$. If you got stuck on (b), you can draw the shape that you think the plot should have and label critical points on the plot.

5. **Performance Measurement Loss (25 points)** Give a short definition of each term below.

   - If there is a mathematical formula associated with the term, write the formula **and** write a one or two sentence explaination of its significance.
   - Otherwise, give a brief (one or two sentences) definition **and** a simple example (one sentence).

   (a) Amdahl's Law

   (b) Computation overhead

   (c) Communication overhead

   (d) Non-parallelizable code

   (e) Speed-up

☺

☺