CMSC330 Spring 2012 Final Exam Solutions

1. (8 pts) Programming languages
   a. (2 pts) Explain why programming languages use types (e.g., int x, char y).
   To specify valid operations for a group of values, and catch errors when the value is used inappropriately.

   b. (2 pts) Explain why programming languages use scopes (e.g., {…}, begin … end).
   To limit the lifetime of a variable name (binding), so the name may be reused.

   c. (2 pts) Explain when programming languages need to use closures.
   When functions are used as return values (downwards funargs), and the returned function *F* can access variables in a enclosing scope from a function that will have already returned by the time *F* is called.

   d. (2 pts) Explain how generic programming in Java is similar to OCaml functions.
   Both use parametric polymorphism to produce code that can work for parameters of differing types.

2. (4 pts) Ruby
   What is the output (if any) of the following Ruby programs? Write FAIL if code does not execute.
   a. (2 pts)

```
a = "Captain America"
if a =~ /([a-z]+)/ then
   puts $1
end
puts $1
```

   # Output =
   aptain
   aptain

   b. (2 pts)

```
a = { }
a["Iron"] = "Man"
puts a["Iron"]
```

   # Output = Man

3. (6 pts) OCaml Types and Type Inference
   a. (2 pts) Give the type of the following OCaml expression
   fun x y z -> [z ; x ; y] **Type = 'a -> 'a -> 'a -> 'a list**

   b. (2 pts) Write an OCaml expression with the following type
   (int -> int list) -> int **Code = fun x -> 2 :: (x 3) ; 4**
   fun x -> match (x 2) with h::t -> h+3

   c. (2 pts) Give the value of the following OCaml expression. If an error exists, describe the error. The function fold is given on the next page.
   fold (fun x y -> x * y) 1 [2;3;4] **Value = 24**

4. (6 pts) OCaml Programming

Using either map or fold and an anonymous function, write a curried function *addFirsts* which when given a list of list of ints *lst,* returns the sum of the 1st elements of all the elements of *lst*. You may assume none of the elements in *lst* are null.

Your function must run in linear time. You may not use any library functions, with the exception of the List.rev function, which reverses a list in linear time. You may not use imperative OCaml (i.e., no ref variables).

```
let rec fold f a l = match l with
        [] -> a
      | (h::t) -> fold f (f a h) t
```

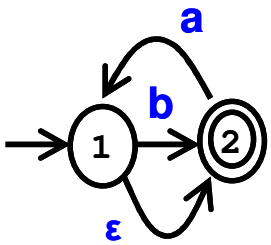
Example:

addFirsts [ ] = 0
addFirsts [[1];[2];[3]] = 6
addFirsts [[1;4];[2;5;6];[3]] = 6

let addFirsts lst = fold (fun a (x::y) -> a+x) 0 lst
let addFirsts lst = fold (fun a b -> match b with x::y -> a+x) 0 lst

5. (8pts) Regular expressions and finite automata

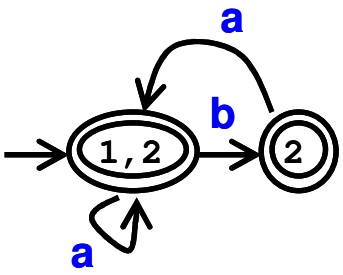
Consider the following NFA.



a. (2 pts) Give a regular expression that accepts the same set of strings as the NFA.

(b|epsilon)(a|ab)* OR (b|epsilon)(a(b|epsilon))*

b. (6 pts) Convert the NFA to a DFA using the subset construction algorithm discussed in class. Be sure to label each state in the DFA with the corresponding state(s) in the NFA.



(8 pts) Context free grammars and parsing

Consider the following grammar

S →	AB	|	Ac
A →	aA	|	epsilon
B →	b	|	epsilon

a. (2 pts) Give a leftmost derivation of the string "ac"

S => Ac => aAc =>ac

b. (4 pts) Calculate FIRST sets for S, A, B

FIRST(S) = { a, b, c, epsilon }
FIRST(A) = { a, epsilon }
FIRST(B) = { b, epsilon }

c. (2 pts) Can the grammar be parsed using a recursive descent parser? Explain. If not, rewrite the grammar so it can be parsed.

No, since FIRST sets of the productions for S are not disjoint (both include { a }). Alternatively, because both productions begin with the same prefix (A).

New grammar =

S →	AL		
L →	B	|	c
A →	aA	|	epsilon
B →	b	|	epsilon

6. (6 pts) Scoping

Consider the following OCaml code.

```
let app f x = f (x+1) ;;
let proc x  = let change z = z+x in app change (x+6) ;;
(proc 2) ;;
```

Value computed by change z is z+x. With static scoping, x = 1st argument for proc. With dynamic scoping, x is 2nd argument app. In both cases, z = 2+6+1 = 9

a. (2 pts) What value is returned by (proc 2) with static scoping? Explain.

11, since value of x in change is the formal parameter x in proc x (i.e., 2).
(proc x:2) -> app change (x:2+6) -> change (z:8+1) -> z+x -> 9+2 -> 11

b. (4 pts) What value is returned by (proc 2) with dynamic scoping? Explain.

17, since value of x in change is the formal parameter x in app f x (i.e., 8).
(proc x:2) -> app change (x:2+6) -> change (z:8+1) -> z+x -> 9+8 -> 17

7. (8 pts) Parameter passing

Consider the following C code.

```
int i = 2;
void foo(int f, int g) {
  f  = f + g;
  g = g + 2;
}
int main( ) {
  int a[] = {1, 1, 1, 1};
  foo(i, a[i-2]);
  printf("%d %d %d %d %d\n", i, a[0], a[1], a[2], a[3]);
}
```

a. (2 pts) Give the output if C uses call-by-value

2 1 1 1 1 // since i & a[0] are unchanged by calling foo

b. (3 pts) Give the output if C uses call-by-reference

3 3 1 1 1 // since i & a[0] are changed by calling foo
// f = i, g = a[0] -> i = i+a[0] ; a[0] = a[0]+2

c. (3 pts) Give the output if C uses call-by-name

3 1 3 1 1 // since i and a[i-2] are changed by calling foo
// f = i, g = a[i-2] -> i = i+a[i-2] ; a[i-2] = a[i-2]+2

8. (4 pts) Garbage collection

Consider the following Java code.

```
class AvengersMovie {
    static Avenger x, y, z;
    private void MoviePlot( ) {
        x = new Avenger (“Iron Man”);          // object 1
        y = new Avenger (“Black Widow”);       // object 2
        z = new Avenger (“Bruce Banner”);      // object 3
        // transformation!
        z = new Avenger(“Hulk”);               // object 4
    }
}
```

a. (2 pts) What object(s) are garbage when MoviePlot ( ) returns? Explain.

Object 3 (Bruce Banner), since it can no longer be accessed.

b. (2 pts) List one advantage and one disadvantage of using garbage collection.

Advantages = less programmer effort, fewer memory errors
Disadvantages = more memory use, extra work during collection.

9. (14 pts) Lambda calculus

Evaluate the following λ-expressions as much as possible.

a. (3 pts) (λx.λy.y x y x) y x z

(λx.λy.y x y x) y x z → (λx.λb.b x b x) y x z → (λb.b y b y) x z → (x y x y) z

b. (3 pts) (λx.λy.λz.y z x) (λc.c) (λa.λb.b a) d

(λx.λy.λz.y z x) (λc.c) (λa.λb.b a) d →
(λy.λz.y z (λc.c)) (λa.λb.b a) d →
(λz.(λa.λb.b a) z (λc.c)) d →
THEN ONE OF
(λa.λb.b a) d (λc.c) → (λb.b d) (λc.c) → (λc.c) d → d
OR
(λz.(λb.b z) (λc.c)) d → (λb.b d) (λc.c) → (λc.c) d → d
OR
(λz.(λb.b z) (λc.c)) d → (λz.(λc.c) z) d → (λz.z) d → d
OR
(λz.(λb.b z) (λc.c)) d → (λz.(λc.c) z) d → (λc.c) d → d

Lambda calculus encodings

c. (8 pts) Using encodings, show succ 2 =>* 3. Show each beta-reduction.
=>* indicates 0 or more steps of beta-reduction

succ = λz.λf.λy.f (z f y)
0 = λf.λy.y
1 = λf.λy.f y
2 = λf.λy.f (f y)
3 = λf.λy.f (f (f y))
4 = λf.λy.f (f (f (f y)))

λz.λf.λy.f (z f y) 2 →
λf.λy.f (2 f y) →
λf.λy.f ((λf.λy.f (f y)) f y) →
λf.λy.f ((λy.f (f y)) y) →
λf.λy.f (f (f y)) →
3

10. (10 pts) Operational semantics

a. (2 pts) In plain English, describe what the following means:

(fun x = (fun y = y x)) 2 → (x:2, λy.y x)

Applying the curried function (fun x = (fun y = y x)) to an argument 2 yields the closure with environment x bound to 2 and the code λy. y x.

OR

Applying a curried function that applies its 2nd argument to its 1st argument to an argument 2 yields a closure that applies its argument to x, which is bound to 2.

b. (8 pts) what does the expression (fun x = (fun y = + x y)) 5 evaluate to in an empty environment? In other words, find a v such that you can prove the following:

• ; (fun x = (fun y = + x y)) 5 → v

Use the operational semantics rules given in class, included here for your reference. Show the complete proof that stacks uses of these rules.

• ; (fun x = (fun y = + x y)) → (•, λx.(fun y = + x y)) // value of closure
• ; 5 → 5 // value of argument
•, x:5 ; (fun y = + x y) → (x:5, (λy.+ x y)) // value of closure body
// evaluated in new env

• ; (fun x = (fun y = + x y)) 5 → (x:5, λy.+ x y)

Number

$$\frac{}{\bullet;\ n \rightarrow n}$$

Addition

$$\frac{A;\ E_1 \rightarrow n \qquad A;\ E_2 \rightarrow m}{A;\ +\ E_1\ E_2 \rightarrow n + m}$$

Identifier

$$\frac{}{A;\ x \rightarrow A(x)}$$

Lambda

$$\frac{}{A;\ \text{fun x = E} \rightarrow (A,\ \lambda x.E)}$$

Function application

$$\frac{A;\ E_1 \rightarrow (A',\ \lambda x.E) \qquad A;\ E_2 \rightarrow v \qquad A,\ A',\ x{:}v;\ E \rightarrow v'}{A;\ (E_1\ \ E_2) \rightarrow v'}$$

11. (12 pts) Multithreading

Consider the following multithreaded Java 1.4 code:

```
class Buffer {
  Buffer ( ) {
    Object buf = null;
    boolean empty = true;
  }
```

```
  void produce(o) {
    synchronize (buf) {
1.    if (!empty) wait( );
2.    empty = false;
3.    notifyAll( );
4.    buf = o;
    }
  }
```

```
  Object consume( ) {
    synchronize (buf) {
5.    if (empty) wait( );
6.    empty = true;
7.    notifyAll( );
8.    return buf;
    }
  }
}
```

```
t1 = Thread.run { produce(1); }
t2 = Thread.run { produce(2); }
t3 = Thread.run { produce(3); }
t4 = Thread.run { x = consume( ); }
t5 = Thread.run { y = consume( ); }
```

Assume thread schedules are given as a list of thread name/line number/range pairs, e.g., (t1, 1), (t4, 5-8), would mean thread 1 executed line 1, followed by thread 4 executing lines 5-8.

For each of the following schedules, determine whether the schedule is possible. If it is, determine what values are assigned to x & y. If the schedule is not possible, explain why.

a. (2 pts) (t3, 1-4), (t5, 5-8), (t2, 1-4), (t4, 5-8), (t1, 1-4)

x = 2, y = 3

Thread t2 puts 3 in buffer, thread t5 reads 3 from buffer, thread t2 puts 2 in buffer, thread t4 reads 2 from buffer, thread t1 puts 1 in buffer.

b. (3 pts) (t2, 1-4), (t3, 1), (t1, 1), (t5, 5-8), (t4, 5), (t1, 2-4), (t4, 6-8), (t3, 2-4)

x = 1, y = 2

Thread t2 puts 2 in buffer, thread t3 waits since buffer is full, thread t1 waits since buffer is full, thread t5 reads 2 from buffer, thread t4 waits since buffer is empty, thread t1 wakes and puts 1 in buffer, thread t4 wakes and reads 1, thread t3 wakes and puts 3 in buffer.

c. (3 pts) (t4, 5), (t5, 5), (t1, 1-4), (t3, 1), (t2, 1), (t5, 6-8), (t4, 6-8), (t3, 2-4)

x = 1, y = 1

Thread t4 waits since buffer is empty, thread t5 waits since buffer is empty, thread t1 puts 1 in buffer, thread t3 waits since buffer is full, thread t2 waits since buffer is full, thread t5 wakes and reads 1 from buffer, thread t4 wakes and reads 1 from buffer (does not check whether buffer is empty since wait is not in a while loop), thread t3 wakes and puts 3 in buffer…

d. (3 pts) (t4, 5), (t1, 1-3), (t4, 6-8), (t1,4), (t3, 1-4), (t5, 5-8), (t2, 1-4)

NOT POSSIBLE

Thread t4 waits since buffer is empty, thread t1 puts 1 in buffer and calls notifyAll, thread t4 wakes and reads 1 from buffer [not possible since t1 has not released lock]…

12. (22 pts) Ruby multithreading

Using Ruby monitors and condition variables, write a Ruby function simulate(M,N) that implements the following simulation of a superhero training facility with M heroes and N training rooms. Each hero arrives and is assigned a number between 0 and M-1, and each training room is assigned a number between 0 and N-1.

Once at the facility, each hero picks a room and trains with another superhero, repeating 10 times. Each room holds up to 2 heroes at a time. Training sessions occur with pairs of superheroes and lasts 0.01 seconds (i.e., call sleep 0.01). Print out a message "Room Z training X & Y" for heroes X and Y training together in each training session in room Z. The choice of a room is not significant. You may use a *getRoom*() function that returns a random room between 0 and N-1 to assign heroes to a room.

Each superhero must be implemented in a separate thread. You must allow pairs of heroes to train at the same time in different rooms (i.e., while one pair is calling sleep 0.01). Once all heroes have finished training, the simulation is complete. You may assume the simulation will automatically be terminated if all active superheroes are waiting in rooms.

You must use monitors to ensure there are no data races, and condition variables to ensure heroes efficiently wait if training rooms are all occupied. The 1st hero entering a training room will also need to efficiently wait for a 2nd hero to join the room (i.e., should wait and only wake up when the 2nd hero enters room). You may only use the following library functions.

Allowed functions:

```
n.times { |i| … }        // executes code block n times, with i = 0…n-1
a = [ ]                  // returns new empty array
a.empty?                 // returns true if array a is empty
a.length                 // returns size of array
a.push(x)                // pushes (adds) x to array a
x = a.pop                // pops (removes) element of a and assigns it to x
a.each { |x| … }         // calls code block once for each element x in a
m = Monitor.new          // returns new monitor
m.synchronize { … }      // only 1 thread can execute code block at a time
c = m.new_cond           // returns conditional variable for monitor
c.wait_while { … }       // sleeps while code in condition block is true
c.wait_until { … }       // sleeps until code in condition block is true
c.broadcast              // wakes up all threads sleeping on condition var c
t = Thread.new {… }      // creates thread, executes code block in new thread
t.join                   // waits until thread t exits
```

Hint: You may want to use an array rooms[] where room[i] = a for room i, where a is another array. a.empty? = true means room is empty. a.push(x) puts a hero x in the room, a[0] returns the 1st hero in the room, a.pop() takes out a hero in the room.

```
require "monitor"
Thread.abort_on_exception = true   # to avoid hiding errors in threads

 def goTrain(me)
      10.times {
        r = getRoom()
        $locks[r].synchronize {
          if $rooms[r].empty?
            $rooms[r].push(me)
            $conds[r].wait_until { $rooms[r].empty? } # woken by partner
          else
            partner = $rooms[r].pop
            sleep 0.01  # train!
            puts "Room #{r} training #{me} & #{partner}"
            $conds[r].broadcast
          end
        }
      }
 end

 def simulate(m,n)
   $numHeroes = m;
   $numRooms = n;
   $rooms = []
   $locks = []
   $conds = []

   # prepare training rooms
   $numRooms.times { |i|
      $rooms[i] = []
      $locks[i] = Monitor.new
      $conds[i] = $locks[i].new_cond
   }

   # start hero training
   threads = []
   $numHeroes.times { |me|
     t = Thread.new { goTrain(me) }
     threads.push(t)
   }
   threads.each { |t| t.join }

 end
```

13. (4 pts) Markup languages

Creating your own XML tags, write an XML document that organizes the following information about the Avengers, a team of superheroes. Captain America started as a fine arts student (really!) and gained his powers from the Super Soldier Serum. The Hulk started as a nuclear physicist and gained his powers through exposure to a nuclear explosion. Iron Man started as an electrical engineer and gains his powers through his mechanical suit.

```
<Avengers>
    <Hero>
        <name>Captain America</name>
        <start>Fine Arts Student</start>
        <powerOrigin>Super Soldier Serum</powerOrigin>
    </Hero>
    <Hero>
        <name>The Hulk</name>
        <start>Nuclear Physicist</start>
        <powerOrigin>Nuclear Explosion</powerOrigin>
    </Hero>
    <Hero>
        <name>Iron Man </name>
        <start>Electrical Engineer</start>
        <powerOrigin>Mechanical Suit</powerOrigin>
    </Hero>
</Avengers>
```