CMSC330 Fall 2011 Final Exam

Name _____

Instructions

- You have 120 minutes for to take this exam.
- This exam has a total of 168 points. An average of 42 seconds per point.
- This is a closed book exam. No notes or other aids are allowed.
- If you have a question, please raise your hand and wait for the instructor.
- Answer essay questions concisely using 1-2 sentences. Longer answers are not necessary and a penalty may be applied.
- In order to be eligible for partial credit, show all of your work and clearly indicate your answers.
- Write neatly. Credit cannot be given for illegible answers.

	Problem	Score	Max Score
1	Programming languages		10
2	Ruby		10
3	Context free grammars & parsing		14
4	OCaml types & type inference		10
5	OCaml higher order & anonymous functions		12
6	Scoping		10
7	Parameter passing		10
8	Lazy evaluation		8
9	Garbage collection		8
10	Lambda calculus & encoding		14
11	Operational semantics		12
12	Java multithreading		14
13	Ruby multithreading		22
14	Polymorphism		8
15	Markup languages		6
	Total		168

1. (12 pts) Programming languages

    a. (3 pts) Briefly define type safety, and describe why it is desirable.

    b. (3 pts) Briefly describe the goal of techniques such as lambda calculus and operational semantics.

    c. (3 pts) Briefly define polymorphism, and explain why it can be a useful feature for a programming language.

    d. (3 pts) Briefly describe how syntax & semantics apply to markup & query languages such as XML and SQL, even though they are not full programming languages.

2. (10 pts) Ruby

Ruby has two classes Array and Hash that are similar but different in important ways. Both are frequently used to store a collection of data, and answer queries on the data (e.g., whether item x is present in the collection). Consider the difference between the following usages of Array and Hash in Ruby.

Given a[x] = y // if a is an array, x is the *index*, y is the *element*
// if a is a hash, x is the *key*, and y is the *value*

The method include?(x) is found in both the Array and Hash classes. For arrays, include?(x) returns true if x is one of the elements of the array. For hashes include?(x) returns if x is a key in the hash.

Consider the following code:

```
a[2] = 3
x = a.include?(2)
y = a.include?(3)
```

a. (2 pts) If the code is preceded by the line a = [ ], what are the values of x & y?

b. (2 pts) If the code is preceded by the line a = { }, what are the values of x & y?

c. (3 pts) What is another important difference between calling a.include?( ) when "a" is an Array vs. a Hash? Hint, it's not whether the answer is true or false.

d. (3 pts) What is a simpler alternative to writing a.include?( ) when "a" is a Hash?

3. (14 pts) Context free grammars and parsing

Consider the following grammar

S →	ABc	|	Bd
A →	a	|	cB
B →	b	|	εpsilon

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

b. (6 pts) Calculate FIRST sets for S, A, B

c. (6 pts) Using pseudocode, write *only* the parse_A function found in a recursive descent parser for the grammar. You may assume the functions parse_S , parse_B already exist.

Use the following utilities:

lookahead	Variable holding next terminal
match (x)	Function to match next terminal to x
error ( )	Reports parse error for input

```
parse_A( ) {    // your code starts here
```

4. (10 pts) OCaml Types and Type Inference

Give the type of the following OCaml expressions:

a. (2 pts) fun x -> x 2 4 Type =

b. (3 pts) fun x -> fun y-> 3 Type =

Write an OCaml expression with the following type:

c. (2 pts) (int * string list) Code =

d. (3 pts) ‘a -> int -> int Code =

5. (12 pts) OCaml higher-order & anonymous functions

Using fold and an anonymous function, write a function *getSeconds* which given an 'a list list returns 'a list, composed of the 1st elements of every list in the original list. The elements should be in the same order as the in the original list. You may assume there all lists in the original list have at least 2 members. 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. Solutions using recursion and/or helper functions will only receive partial credit.

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

Examples:

getSeconds [] = []
getSeconds [[1;2]] = [2]
getSeconds [[1;2];[3;4]] = [2;4]
getSeconds [[1;2];[3;4];[5;6;7]] = [2;4;6]

6. (10 pts) Scoping

Consider the following OCaml code.

```
let app f x = f x ;;
let proc x  = let change z = z+x in app change (x+5) ;;
(proc 3) ;;
```

a. (2 pts) What is the order the functions app, proc, and change are invoked?

b. (4 pts) What value is returned by (proc 3) with static scoping? Explain.

c. (4 pts) What value is returned by (proc 3) with dynamic scoping? Explain.

7. (10 pts) Parameter passing

Consider the following C code.

```
int i = 1;
void foo(int f, int g) {
  f  = f + g;
  g = g + 2;
}
int main( ) {
  int a[] = {3, 5, 7, 9};
  foo(i, a[i-1]);
  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

b. (4 pts) Give the output if C uses call-by-reference

c. (4 pts) Give the output if C uses call-by-name

8. (8 pts) Lazy evaluation
   a. (3 pts) Explain why lazy evaluation allows some programs to successfully execute that would not execute using eager evaluation.

   b. (5 pts) Rewrite the following code (using thunks) so that *foo* evaluates its argument only when it is used, even though OCaml uses call-by-value.

```
let foo f = [f] ;;
foo 1  ;;
```

9. (8 pts) Garbage collection

Consider the following Java code.


```
class Inception {
        static DreamLayer current, up1, up2;
        private void MoviePlot( ) {
                up2 = new DreamLayer ("van");              // object 1
                up1 = new DreamLayer ("hotel");            // object 2
                current = new DreamLayer ("fortress");     // object 3
                // …dreamKick…
                current = up1;
                up1 = up2;
        }
}
```

   a. (4 pts) What object(s) are garbage when MoviePlot ( ) returns? Explain.

   b. (4 pts) List one advantage and one disadvantage of using garbage collection.

10. (14 pts) Lambda calculus

Evaluate the following λ-expressions as much as possible.

a. (3 pts) (λx.λy.x y) y z x

b. (3 pts) (λx.λy.y x) a (λz.b z) c

Lambda calculus encodings

c. (8 pts) Using encodings, show 1*3 =>* 3. Show each beta-reduction.

=>* indicates 0 or more steps of beta-reduction

M * N = λx.(M (N x))
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)))

11. (12 pts) Operational semantics

a. (4 pts) In plain English, describe what the following means:

y:1 ; fun x = x → (y:1, λx.x)

b. (8 pts) In an empty environment, what does the expression (fun x = x+2) evaluate to? In other words, find a v such that you can prove the following:

• ; (fun x = x+2) → v

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

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

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

12. (14 pts) Multithreading

Consider the following attempt to implement producer/consumer pattern w/ Java 1.4.

```
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;       // also releases lock
       }
    }
}
t1 = Thread.run { produce(1); }
t2 = Thread.run { produce(2); }
t3 = Thread.run { x = consume( ); }
t4 = Thread.run { y = consume( ); }
```

In the following, give schedules as a list of thread name/line number/range pairs, e.g., (t1, 1-4), (t2, 1), (t3, 5-8). For instance, one schedule under which x=1 and y=2 is (t1, 1-4), (t3, 5-8), (t2, 1-4), (t4, 5-8)

a. (2 pts) Give a schedule under which x = 2 and y = 1.

b. (4 pts) Give a schedule under which x = 2 and y = 2, or argue that no such schedule is possible.

c. (8 pts) Explain why the given Java code allows data races and why deadlock may occur.

13. (22 pts) Ruby multithreading

Using Ruby monitors and condition variables, write a Ruby function simulate(M,N) that implements the following simulation of a dance club. M girls and N guys arrive at a club. Each guy is assigned a number between 0 and N-1, and each girl is assigned a number between 0 and M-1. Once at the club, each girl dances 10 times, each time picking any guy who is not currently dancing with another girl. Each dance lasts 0.01 seconds in real time (i.e., call sleep 0.01). Print out a message "X dancing with Y" for girl X and guy Y at the start of each dance. The action for each girl must be executed in a separate thread. You must allow multiple couples to dance at the same time (i.e., while calling sleep 0.01). Once all girls have finished dancing, the simulation is complete.

You must use monitors to ensure there are no data races, and condition variables to ensure girls efficiently wait if all guys are currently dancing. 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.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
```

14. (8 pts) Polymorphism

a. (4 pts) Does OCaml uses ad hoc or parametric polymorphism? Briefly explain.

b. (4 pts) Briefly explain why "? extends A" is needed for Java generics.

15. (6 pts) Markup languages

Creating your own XML tags, write an XML document that organizes the following information about turtles. Sea turtles live in the ocean and can grow to 2000 pounds. Tortoises live on land and can grow to 660 pounds. Terrapins live in rivers and can grow to 130 pounds.