CMSC330 Fall 2014 Final Exam

Name:

_____

Discussion Time	**10am**	**11am**	**noon**	**1pm**	**2pm**	**3pm**
TA Name (circle):	**Casey Ilse**	**Casey Daniel**	**Xuefang Yogarshi**	**Xuefang**	**Ian**	**Ian**

Instructions

- Do not start this test until you are told to do so!
- You have 120 minutes to take this exam.
- This exam has a total of 120 points, so allocate 60 seconds for each point.
- This is a closed book exam. No notes or other aids are allowed.
- For partial credit, show all of your work and clearly indicate your answers.
- Write neatly. Credit cannot be given for illegible answers.

	Problem	Score
1	Programming Languages	/10
2	Ruby & OCaml	/8
3	Scoping	/5
4	Parameter Passing	/5
5	Lazy Evaluation	/6
6	Garbage Collection	/4
7	Multithreading	/6
8	Ruby Multithreading	/18
9	Prolog	/12
10	Prolog Programming	/18
11	Lambda Calculus	/12
12	Operational Semantics	/10
13	Markup Languages	/4
	Total	/120

HONOR PLEDGE: I pledge on my honor that
I have not given or received any unauthorized
assistance on this assignment/ examination. SIGNATURE: _____

1. (10 pts) Programming languages (PL)

   (1 pt each) For the following multiple choice questions, circle the letter on the right corresponding to the best answer to each question. A question may have multiple answers.

   a. Which following term(s) is *not* a PL programming paradigm? A B C D
      A) imperative B) functional C) logical D) hierarchical

   b. OCaml module signatures are used to specify which components of a module are accessible from the outside. They are similar to which of the following?
      A) Java classes A B C D
      B) Java interfaces
      C) .h files in C
      D) .c files in C

   c. Which PL(s) has a strong type system with static types? A B C D
      A) Ruby B) Java C) C D) Prolog

   d. Which PL(s) has a weak type system with static types? A B C D
      A) Ruby B) Java C) C D) Prolog

   e. Which following term(s) is *not* a garbage collection (GC) technique? A B C D
      A) mark & sweep B) stop & copy C) malloc & free D) reference counting

   f. Which code example(s) below uses parametric polymorphism? A B C D
      A) a = [1,"2"] B) puts x+y C) fun x y -> x y D) fun x y -> x+y

   g. Which code example(s) below uses ad hoc polymorphism? A B C D
      A) a = [1,"2"] B) puts x+y C) fun x y -> x y D) fun x y -> x+y

   h. Which Java statement(s) is illegal given class B extends class A? A B C D
      A) A a = new B( );
      B) A[ ] a = new B[ ];
      C) Set<A> a = new Set<B>;
      D) Set<? extends A> a = new Set<B>;

   i. Which PL is considered an ancestor to C, Java, and Ruby? A B C D
      A) Algol B) Cobol C) Fortran D) Lisp

   j. Which feature(s) is *not* a mistake made by a past PL? A B C D
      A) Spaces in variable names
      B) Non-reserved keywords
      C) Call by reference
      D) 2-digit representation of year

2. (8 pts) Ruby & OCaml

   a. (2 pts) What is the output (if any) of the following Ruby programs? If an error exists, describe the error.

```
a = { }
a[“Spade”] = [ ]
a[“Spade”][“Club”] = “Heart”
a[1][“Heart”] = “Diamond”
puts “Draw #{a[“Spade”][“Club”]}”
```

   OUTPUT =

   b. (2 pts) Give the type of the following OCaml expression:

   (fun x → (let y = 1 in x+y)) **Type =**

   c. (2 pts) Write an OCaml expression with the following type

   (int -> int) list -> int **Code =**

   d. (2 pts) Give the value of the following OCaml expression. If an error exists, describe the error.

   (fun x → (let y = 1 in x+y)) 2 **Value =**

3. (5 pts) Scoping

Consider the following OCaml code.

```
let app f x = let y = 4 in (f x) – x ;;
let proc y  = let mult x = x * y in app mult (y+3) ;;
(proc 2) ;;
```

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

b. (3 pts) What value is returned by (proc 2) with dynamic scoping? Explain.

4. (5 pts) Parameter passing

Consider the following C code.

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

a. (1 pts) Give the output if C uses call-by-value

b. (2 pts) Give the output if C uses call-by-reference

c. (2 pts) Give the output if C uses call-by-name

5. (6 pts) Lazy evaluation

Rewrite the following OCaml code using thunks so that foo uses lazy evaluation.

```
let foo x = x – 2 ;;
foo (foo 4) ;;
```

6. (4 pts) Garbage collection

Consider the following Java code.

```
class OnlinePoker {
        static Company x, y, z;
        private void CorporateMergers( ) {
                x = new Company ("Poker Stars");          // object 1
                y = new Company ("Party Poker");          // object 2
                z = new Company ("Full Tilt Poker");      // object 3
                z = x;   // Poker Stars buys Full Tilt Poker!
                // bwin merges with Party Poker!
                y = new Company ("bwin");                 // object 4

        }
}
```

What object(s) are garbage when CorporateMergers( ) returns? Explain.

7. (6 pts) Multithreading

Consider the preceding multithreaded Java 1.4 code. Assume there are multiple producer and consumer threads being executed in the program, but only a *single* Buffer object. Questions about the "last statement executed" by a thread refer to the most recently executed statement by that thread at some arbitrary time during the program execution. It does not mean the last statement executed by a thread before the thread exits. If a situation is possible, you need to give an example of how it is possible (e.g., thread x gets to statement a, then thread y gets to statement b). If a situation is not possible, you need to explain why.

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

```
  void produce(o) {
    synchronize (this) {
1.    if (buf) wait( );
2.    buf = o;
3.    notifyAll( );
    } }
```

```
  Object consume( ) {
    synchronize (this) {
4.    while (!buf) wait( );
5.    Object tmp = buf;
6.    notifyAll( );
7.    buf = null;
8.    return tmp;
    } } }
```

a. (2 pts) Is it possible given 3 threads x, y, and z for the last statement executed by thread x to be statement 5, thread y to be statement 4, and thread z to be statement 1 in the code above? Explain your answer.

b. (2 pts) Is it possible in the code above for two threads x & y calling consume( ) to have consume( ) return null for thread x? Assume produce(o) is never called with o == null. Explain your answer.

c. (2 pts) Is it possible in the code above for two threads x & y calling produce(o) to have x overwrite the value y assigns to buf? Explain your answer.

8. (18 pts) Ruby multithreading

Using Ruby monitors and condition variables, write a Ruby function simulate(*p,d,t*) that implements the following simulation of a poker room with *p* players, *d* dealers, and *t* tables. Players, dealers, and tables are assigned IDs starting from 0 and ascending by 1.

Each poker table holds 8 players and 1 dealer. Poker tables are initially empty. Once a dealer sits at a table, players sit until the table is full. The dealer then hosts a tournament, calling sleep 0.01 and printing out the message "Table: *x y*" for table x and tournament number y, where y start at 1 and ascends by 1. Once the tournament is over, players leave. The dealer leaves after all players leave, and the process is repeated. You must use the rand(*t*) function to choose a table number between 0 and *t*-1.

Each player and dealer must be implemented in a separate thread. You must allow tournaments at different tables to take place in parallel. Each player and dealer participates in one tournament. Once all players have participated in a tournament, the simulation is complete. You may assume the simulation will automatically be terminated if all remaining players are waiting at poker tables for tournaments to begin.

You must use monitors to ensure there are no data races, and condition variables to ensure dealers and players wait efficiently when needed. Use multiple conditional variables for efficiency & full credit. You may 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.size                   // returns size of array
a.push(x)                // pushes (adds) x to end of array a
x = a.pop                // pops (removes) element of a from end & assigns to x
stmt until (p)           // execute stmt while p is false
a.each { |x| … }         // calls code block once for each element x in a
rand(n)                  // returns an integer value between 0 and n-1
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 = Thread.new(x) { |x| … }   // executes code block in new thread with arg x
t.join                   // waits until thread t exits
```

The following code is a semi-sequential version of the simulation. You should be able to implement your function by adding threads, synchronization, and conditional variables to it.

```
class PokerTable

def initialize(t)
  @tabNum = t
  @players = []
  @state = 0     #  waiting for dealer to arrive
end

def dealTourney(id)
  until (@state == 0) #  waiting for dealer to arrive
  @state = 1 # waiting for players to arrive
  until (@players.size == 8 }
  puts "Table: #{@tabNum} #{$tournamentNum}"
  $tournamentNum += 1
  sleep 0.01
  @state = 2 # waiting for players to leave
  until (@players.size == 0)
  @state = 0 # waiting for dealer to arrive
end
```

```
def playTourney(id)
  until ((@state == 1) && (@players.size < 8))
  @players.push(id)
  until (@state == 2) # waiting for players to leave
  @players.delete(id)
end

end # end class PokerTable

def simulate(p,d,t)
  $tournamentNum = 1
  tables = []
  t.times  { |x| tables[x] = PokerTable.new(x) }
  d.times { |x| tables[rand(t)].dealTourney(x) }
  p.times { |x| tables[rand(t)].playTourney(x) }
end

# simulate 100 players, 20 dealers, 5 tables
simulate(100,20,5)
```

9. (12 pts) Prolog

The following Prolog code (written by a newbie) attempts to recognize various poker hands. List all answers returned by the following queries.

```
pair(N) :- hand(N,A,A,B,_,_), A\=B.
twoPair(N) :- hand(N,A,A,B,B,C).
trips(N) :- hand(N,A,A,A,B,C).
trips(N) :- hand(N,A,B,C,C,C).
fullHouse(N) :- pair(N), !, trips(N).
quads(N) :- A\=B, hand(N,A,A,A,A,B).

hand(h1,2,2,4,5,7).
hand(h2,5,5,7,7,7).
hand(h3,4,4,4,9,jack).
hand(h4,ace,ace,ace,ace,king).
```

a. (1 pts) ?- pair(N).

b. (1 pts) ?- twoPair(N).

c. (2 pts) ?- trips(N).

d. (2 pts) ?- fullHouse(N).

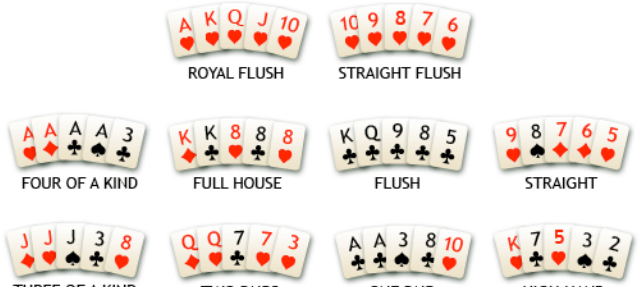
e. (1 pts) ?- quads(N).

```
foo([X|T],R) :- foo(T,R).
foo([X,X|T],X).
```

f. (1 pts) ?- foo([1,1,2],A).

g. (2 pts) ?- foo([1,1,2,2],A).

h. (2 pts) ?- foo([1,2,2,2,3,3],A)



10. (18 pts) Prolog programming

Write a prolog function dealTwo(A,X,Y,R) that given a list A, returns two elements X and Y from A and the remaining elements of A in R. X and Y may not be the same element, though they may have the same value if A contains duplicate elements,. Additional requests to dealTwo should eventually return all possible pairs of elements from list A (in any order), subject to the conditions above. You may use the operators !, =, \=, \+, is, +, -, [H|T], [H1,H2|T], etc. You may not use semicolon ;. You do not need to worry about efficiency.

Examples:

?- dealTwo([],X,Y,R). false. ?- dealTwo([1],X,Y,R). false.	?- dealTwo([1,2],X,Y,R). X=1, Y=2, R=[]; X=2, Y=1, R=[]. ?- dealTwo([1,1],X,Y,R). X=1, Y=1, R=[]; X=1, Y=1, R=[].	?- dealTwo([1,2,3],X,Y,R). X=1, Y=2, R=[3]; X=1, Y=3, R=[2]; X=2, Y=1, R=[3]; X=2, Y=3, R=[1]; X=3, Y=1, R=[2]; X=3, Y=2, R=[1].

11. (14 pts) Lambda calculus

Evaluate the following λ-expressions as much as possible.

a. (2 pts) (λx.λy.λz.y z x) z x y

b. (4 pts) (λx.λy.x (x y)) (λz.y z) x

Lambda calculus encodings

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

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

M * N = λx.(M (N x))
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)))

12. (10 pts) Operational semantics

What does the expression (fun x → (let y = 1 in x+y)) 2 evaluate to in an empty environment? In other words, find a v such that you can prove the following:

- ; (fun x → (let y = 1 in x+y)) 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. *Put a number label (e.g., #1, #2, etc.) next to each hypothesis to indicate the order they are used in your proof.*

$A(x) = v$	-
$A; x \Rightarrow v$	$A; n \Rightarrow n$

$A; E_1 \Rightarrow v_1$ $\quad A; E_2 \Rightarrow v_2$
$A; E_1 \text{ op } E_2 \Rightarrow v_1 \text{ op } v_2$

–
$A;$ `fun` $x \rightarrow E \Rightarrow (A, \lambda x.E)$

$A; E_1 \Rightarrow v_1$ $A, x{:}v_1; E_2 \Rightarrow v_2$
$A;$ `let` $x =$ E_1 `in` $E_2 \Rightarrow v_2$

$A; E_1 \Rightarrow (A', \lambda x.E)$ $A; E_2 \Rightarrow v_2$ $A', x{:}v_2; E \Rightarrow v$
$A; E_1\ E_2 \Rightarrow v$

13. Markup languages

Creating your own XML tags, write an XML document that organizes the following information about nicknames for 2-card starting hands in Texas Hold'em, a popular poker game. Pocket Rockets is two Aces. King Kong is two Kings. Big Slick is an Ace and a King (in the same suit). Pocket Rockets and King Kong are *pairs*. Big Slick is a *suited connector*.