

CMSC330 Fall 2014 Final Exam Solutions

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 **B** **C** D
A) Java classes
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;`
D) `Set<? extends A> a = new Set;`
- 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 = TypeError: can't convert String into Integer

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

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

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

(int -> int) list -> int **Code = fun (h::t) -> 1+(h 2)**

- 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**

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.

(proc 2) => mult x = x*2 => app mult (2+3) => (mult 5) - 5 => (5*2) - 5 = 5.

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

(proc 2) => app mult (2+3) => y=4 => (mult 5) - 5 => (5*4) - 5 = 15.

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 | 1,1,1,1,1 |
| b. (2 pts) Give the output if C uses call-by-reference | 0,1,1,2,1 |
| c. (2 pts) Give the output if C uses call-by-name | 0,1,2,1,1 |

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) ;;
```

```
let foo x = (x ()) - 2 ;;
foo (fun () -> (foo (fun () -> 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.

Objects 2 & 3 are garbage since they are no longer reachable.

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.

<pre>class Buffer { Buffer () { Object buf = null; } }</pre>	<pre>void produce(o) { synchronize (this) { 1. if (buf) wait(); 2. buf = o; 3. notifyAll(); } }</pre>	<pre>Object consume() { synchronize (this) { 4. while (!buf) wait(); 5. Object tmp = buf; 6. notifyAll(); 7. buf = null; 8. return tmp; } }</pre>
--	---	---

- 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.

Yes, since thread y and thread z could have both released the lock for the buffer by calling wait.

- 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.

No, since wait is called within a while loop thread x may not continue past the wait until buf is not null.

- 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.

Yes, since wait is not called within a while loop thread x may continue past wait even though buf is not null.

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
```

Hint: You can start by modifying the following sequential version of the simulation:

<pre> 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} #{\$tourneyNum}" \$tourneyNum += 1 sleep 0.01 @state = 2 # waiting for players to leave until (@players.size == 0) @state = 0 # waiting for dealer to arrive end </pre>	<pre> 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) \$tourneyNum = 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) </pre>
--	--

```
require "monitor.rb"
```

```
class PokerTable
```

```
def initialize(t)
```

```
  @tabNum = t
```

```
  @players = []
```

```
  @state = 0 # 0 - waiting for dealer, 1 - waiting for players, 2 - emptying table
```

```
  @myLock = Monitor.new
```

```
  @dealerCondition = @myLock.new_cond
```

```
  @playerCondition = @myLock.new_cond
```

```
end
```

```

def dealTourney(id)
  @myLock.synchronize {
    @dealerCondition.wait_until { @state == 0 }
    @state = 1
    @playerCondition.broadcast
    @dealerCondition.wait_until { @players.size == 8 }
    $tLock.synchronize {
      puts "Table: #{@tabNum} #{$tourneyNum}"
      $tourneyNum += 1
    }
  }
  sleep 0.01
  @myLock.synchronize {
    @state = 2
    @playerCondition.broadcast
    @dealerCondition.wait_until { @players.size == 0 }
    @state = 0
    @dealerCondition.broadcast
  }
end

def playTourney(id)
  @myLock.synchronize {
    @playerCondition.wait_until { (@state == 1) && (@players.size < 8) }
    @players.push(id)
    @dealerCondition.broadcast
    @playerCondition.wait_until { (@state == 2) }
    @players.delete(id)
    @dealerCondition.broadcast
  }
end

def simulate(p,d,t)
  $tourneyNum = 0
  $tLock = Monitor.new
  tables = []
  players = []
  t.times { |x| tables[x] = PokerTable.new(x) }
  p.times { |x| players[x] = Thread.new(x) { |y| tables[rand(t)].playTourney(y) } }
  d.times { |x| Thread.new(x) { |y| tables[rand(t)].dealTourney(y) } }
  p.times { |x| players[x].join }
end
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.

<pre> 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). </pre> <p>a. (1 pts) ?- pair(N).</p> <p>N = h1; N = h2.</p> <p>b. (1 pts) ?- twoPair(N).</p> <p>N = h2; N = h4.</p> <p>c. (2 pts) ?- trips(N).</p> <p>N = h3; N = h4; N = h2.</p> <p>d. (2 pts) ?- fullHouse(N).</p> <p>false.</p> <p>e. (1 pts) ?- quads(N).</p> <p>false.</p>	<pre> foo([X T],R) :- foo(T,R). foo([X,X T],X). </pre> <p>f. (1 pts) ?- foo([1,1,2],A).</p> <p>A = 1.</p> <p>g. (2 pts) ?- foo([1,1,2,2],A).</p> <p>A = 2; A = 1.</p> <p>h. (2 pts) ?- foo([1,2,2,2,3,3],A)</p> <p>A = 3; A = 2; A = 2.</p>
---	---

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:

<pre>?- dealTwo([],X,Y,R). false. ?- dealTwo([1],X,Y,R). false.</pre>	<pre>?- dealTwo([1,2],X,Y,R). X=1, Y=2, R=[]; X=2, Y=1, R=[].</pre> <pre>?- dealTwo([1,1],X,Y,R). X=1, Y=1, R=[]; X=1, Y=1, R=[].</pre>	<pre>?- 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].</pre>
---	---	---

`dealOne([H|T],H,T).`

`dealOne ([H|T],X,[H|Y]) :- dealOne (T,X,Y).`

`dealTwo(A,X,Y,R2) :- dealOne (A,X,R1), dealOne (R1,Y,R2).`

11. (14 pts) Lambda calculus

Evaluate the following λ -expressions as much as possible.

a. (2 pts) $(\lambda x.\lambda y.\lambda z.y z x) z x y$

$$(\lambda a.\lambda b.\lambda c.b c a) z x y \Rightarrow (\lambda b.\lambda c.b c z) x y \Rightarrow (\lambda c.x c z) y \Rightarrow x y z$$

b. (4 pts) $(\lambda x.\lambda y.x (x y)) (\lambda z.y z) x$

Sample Answers

$(\lambda x.\lambda y.x (x y)) (\lambda z.y z) x \Rightarrow$ $(\lambda a.\lambda b.a (a b)) (\lambda z.y z) x \Rightarrow$ $(\lambda b. (\lambda z.y z) ((\lambda z.y z) b)) x$ $(\lambda z.y z) ((\lambda z.y z) x)$ $(\lambda z.y z) (y x)$ $y (y x)$	$(\lambda x.\lambda y.x (x y)) (\lambda z.y z) x \Rightarrow$ $(\lambda a.\lambda b.a (a b)) (\lambda z.y z) x \Rightarrow$ $(\lambda b. (\lambda z.y z) ((\lambda z.y z) b)) x$ $(\lambda b. (\lambda z.y z) (y b)) x$ $(\lambda b. y (y b)) x$ $y (y x)$	$(\lambda x.\lambda y.x (x y)) (\lambda z.y z) x \Rightarrow$ $(\lambda a.\lambda b.a (a b)) (\lambda z.y z) x \Rightarrow$ $(\lambda b. (\lambda z.y z) ((\lambda z.y z) b)) x$ $(\lambda b. y ((\lambda z.y z) b)) x$ $(y ((\lambda z.y z) x))$ $y (y x)$
---	---	--

Lambda calculus encodings

c. (8 pts) Using encodings, show $3*1 \Rightarrow^* 3$. Show each beta-reduction.

\Rightarrow^* indicates 0 or more steps of beta-reduction

$M * N = \lambda x.(M (N x))$ $0 = \lambda f.\lambda y.y$ $1 = \lambda f.\lambda y.f y$ $2 = \lambda f.\lambda y.f (f y)$ $3 = \lambda f.\lambda y.f (f (f y))$ $4 = \lambda f.\lambda y.f (f (f (f y)))$
--

Sample Answers

$3*1 \Rightarrow \lambda x.(M (N x))$ $\Rightarrow \lambda x.(3 (1 x))$ $\Rightarrow \lambda x.(3 ((\lambda f.\lambda y.f y) x))$ $\Rightarrow \lambda x.(3 (\lambda y.x y))$ $\Rightarrow \lambda x.((\lambda f.\lambda y.f (f (f y))) (\lambda y.x y))$ $\Rightarrow \lambda x.(\lambda y.((\lambda y.x y) ((\lambda y.x y) ((\lambda y.x y) y))))$ $\Rightarrow \lambda x.(\lambda y.((\lambda y.x y) ((\lambda y.x y) (x y))))$ $\Rightarrow \lambda x.(\lambda y.((\lambda y.x y) (x (x y))))$ $\Rightarrow \lambda x.(\lambda y.(x (x (x y))))$ $\Rightarrow 3$	$3*1 \Rightarrow \lambda x.(M (N x))$ $\Rightarrow \lambda x.(3 (1 x))$ $\Rightarrow \lambda x.((\lambda f.\lambda y.f (f (f y))) (1 x))$ $\Rightarrow \lambda x.(\lambda y.((1 x) ((1 x) ((1 x) y))))$ $\Rightarrow \lambda x.(\lambda y.((1 x) ((1 x) (((\lambda f.\lambda y.f y) x) y))))$ $\Rightarrow \lambda x.(\lambda y.((1 x) ((1 x) (((\lambda y.x y) y))))$ $\Rightarrow \lambda x.(\lambda y.((1 x) ((1 x) (x y))))$ $\Rightarrow \lambda x.(\lambda y.((1 x) (((\lambda f.\lambda y.f y) x) (x y))))$ $\Rightarrow \lambda x.(\lambda y.((1 x) ((\lambda y.x y) (x y))))$ $\Rightarrow \lambda x.(\lambda y.((1 x) (x (x y))))$ $\Rightarrow \lambda x.(\lambda y.(((\lambda f.\lambda y.f y) x) (x (x y))))$ $\Rightarrow \lambda x.(\lambda y.(((\lambda y.x y) (x (x y))))$ $\Rightarrow \lambda x.(\lambda y.((x (x (x y))))$ $\Rightarrow 3$
--	---

12. (10 pts) Operational semantics

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

$$\bullet ; (\text{fun } x \rightarrow (\text{let } y = 1 \text{ in } x+y)) \ 2 \Rightarrow v$$

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.*

#7 $\bullet, x:2, y:1 ; x \Rightarrow 2$

#8 $\bullet, x:2, y:1 ; y \Rightarrow 1$

#5 $\bullet, x:2 ; 1 \Rightarrow 1$

#6 $\bullet, x:2, y:1 ; x+y \Rightarrow 3$

#2 $\bullet ; (\text{fun } x \rightarrow (\text{let } y = 1 \text{ in } x+y)) \Rightarrow (\bullet, \lambda x. (\text{let } y = 1 \text{ in } x+y))$

#3 $\bullet ; 2 \Rightarrow 2$

#4 $\bullet, x:2 ; (\text{let } y = 1 \text{ in } x+y) \Rightarrow 3$

#1 $\bullet ; (\text{fun } x \rightarrow (\text{let } y = 1 \text{ in } x+y)) \ 2 \Rightarrow 3$

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

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

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

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

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

13. (4 pts) 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*.

```
<starting Hold'em hands>
  <hand type="pair">
    <name>Pocket Rockets</name>
    <cards>Ace,Ace</cards>
  </hand>
  <hand type="pair">
    <name>King Kong</name>
    <cards>King,King</cards>
  </hand>
  <hand type="suited connector">
    <name>Big Slick</name>
    <cards>Ace,King</cards>
  </hand>
</starting Hold'em hands>
```