

# CMSC330 Spring 2015 Final Exam

## 9:30am/11:00am/12:30pm

Name:

UID:

Discussion Time (circle one): 10am 11am 12pm 1pm 2pm 3pm  
Discussion TA (circle one): Amelia Casey Chris Mike Elizabeth Eric Tommy

### Instructions

- The exam has 3 pages; make sure you have them all.
- Do not start this test until you are told to do so!
- You have 120 minutes to take this midterm.
- This exam has a total of 130 points, so allocate 55 seconds for each point.
- This is a closed book exam. No notes or other aids are allowed.
- Answer essay questions concisely in 2-3 sentences. Longer answers are not needed.
- 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	PL Concepts	/22
2	Lambda Calculus	/6
3	OCaml Types	/9
4	OCaml Execution	/9
5	OCaml Programming	/14
6	Ruby Programming	/15
7	Prolog Execution	/15
8	Prolog Programming	/10
9	Multithreading in Ruby	/18
10	Regexps, FAs, CFGs	/12
	<b>TOTAL</b>	<b>/130</b>

**1. PL Concepts (22 pts)**

- a. (3 pts) Describe how the execution of the following OCaml program would differ with Call-by-Name parameter passing, compared to Call-by-Value:

```
let f x y =  
  if x>0 then (y,y) else (1,1);;  
f 1 (print_string "hello"; 2);;
```

- b. (3 pts) Circle the following memory errors that can happen in a Ruby program.

*Dangling pointer dereference*    *Double-free*    *Memory leak*

- c. (4 pts) Circle the following languages below that have (mostly) static typing.

*OCaml*    *Ruby*    *Prolog*    *Java*

- d. (4 pts) Draw or describe the closure returned by calling  $(g \ 0)$ , where  $g$  is defined as follows:

```
let g n =  
  let r = ref 0 in  
  let f m = !r + n + m in  
  let x = f 0 in  
  f
```

- e. (2 pts) True or false: if  $e_1$  and  $e_2$  are physically equal, then they are also structurally equal. (You may optionally justify your answer.)

- f. (6 pts) Consider the following Java code. Circle (and label) an example each of the use of *ad hoc polymorphism*, *subtype polymorphism*, and *parametric polymorphism* (generics) in the code.

```
public interface Foo {
    public void shout();
}

public class Bar implements Foo {
    public void shout() { System.out.println("foo!"); }
    public void shout(String exclaim) {
        System.out.println("foo says "+exclaim+"!");
    }

    public static <T> int count(Collection<T> stuff) {
        int count = 0;
        for (T o : stuff) { count++; }
        return count;
    }

    public static void main(String args[]) {
        Collection<Foo> c = new ArrayList<Foo>(10);
        Bar b = new Bar();
        b.shout();
        b.shout("again");
        c.add(b);
        System.out.println(count(c));
    }
}
```

**2. Lambda Calculus (6 pts).**

a. (3 pts) What do you get if you fully reduce the following lambda calculus term?

$(\lambda x.x) (\lambda x.\lambda y.x) w z$

b. (3 pts) Which of the following pairs of terms are alpha equivalent (i.e., you can alpha convert one into the other)?

i.  $(\lambda x.\lambda y.x y) y z$                        $(\lambda x.\lambda y.x y) w z$

ii.  $(\lambda x.\lambda y.x y) y z$                        $(\lambda x.\lambda w.x w) y z$

iii.  $(\lambda x.\lambda y.x y) y z$                        $(\lambda x.\lambda y.x x) y z$

**3. OCaml typing (9 pts).**

a. (3 pts) What is the type of f in the following definition?

```
let f x = (x 1)+1
```

b. (3 pts) What is the type of the following OCaml expression?

```
(fun x -> fun y -> y::x)
```

c. (3 pts) What is the type of f in the following definition?

```
let f x a b = if x>0 then (a x) else (b x)
```

4. **OCaml execution (9 pts):** What is the value of the variable `result` after executing the following code? If an error will occur, say what the problem is. (Note there are no syntax errors in these programs.)

a. (3 pts)

```
let rec aux m x =
  match m with
  | [] -> []
  | h::t ->
    let t' = if h > x then [] else aux t x in
    h::t'
let result = aux [1;2;4;7;7;0] 4;;
```

b. (3 pts)

```
type t = Int of int | String of string
let f x =
  match x with
  | Int i -> string_of_int (i+1)
  | String _ -> "2";;
let result = (f (String "hi"), f (Int 1))
```

c. (3 pts)

```
let co f g x =
  let z = f x in
  g z;;
let result = co (fun x -> x + 1) (fun y -> 0) 2
```

5. **OCaml coding (14 pts).** For the following you may write helper functions if you like. You may also use standard library functions, e.g., from the `List` and `Pervasives` modules, unless otherwise noted.

a. (4 pts) Write a function `flat` of type

```
( 'a -> 'b -> 'c ) -> ( 'a * 'b ) -> 'c.
```

This function takes a curried function as its argument and returns a function that takes the arguments as a tuple instead. So:

```
let f = flat (fun x y -> x + y);; // f has type int*int -> int
let result = f (1,2);; // result contains 3
```

The answer is actually very simple (one line): Don't overthink it!

b. (10 pts) **Either do this problem or the next one. If you do both we'll average your scores.** Write a function `splast` of type `'a list -> 'a list * 'a` which when given a non-empty list `l` will return a pair where the first component of the pair is the same as the input list but missing the last element, and the second component of the pair is the last element of the input list.

Examples:

```
splast [0;3] = ([0],3)
```

```
splast [1;2;3;4] = ([1;2;3],4)
```

- c. (10 pts) **Either do this problem or the previous one. If you do both we'll average your scores.** Using either map or fold (code below) and an anonymous function, write a curried function `partialProduct` which when given a list of ints `lst` and an integer `x`, returns the product of each element in `lst` which is greater than `x`. The partial product of an empty list is 1. (If you use a recursive function and not map/fold you can earn at most 80% partial credit).

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

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

*Examples:*

```
partialProduct [ ] 0 = 1
```

```
partialProduct [ ] 5 = 1
```

```
partialProduct [0;3] 1 = 3
```

```
partialProduct [1;2;3;4] 2 = 12
```

6. **Ruby execution (15 pts).** Give the output of executing the following programs. If executing the program produces a failure of some sort, write the output up to that failure, and then write FAIL.

a. (3 pts)

```
a = "CMSC330"  
if a =~ /([\d]+)/ then  
  puts $1  
end
```

**Answer:**

b. (3 pts)

```
a = { }  
a[1] = "me"  
puts a[0]
```

**Answer:**

c. (3 pts)

```
x = [3, 5, 8].map {|num| num * 2}  
puts x
```

**Answer:**

d. (3 pts)

```
x = [{"harry",6}, {"tom",3}, {"jack",9}]  
x.sort! {|x,y| x[1] <=> y[1]}  
puts x
```

**Answer:**

e. (3 pts)

```
h = Hash.new(100)  
h["a"]=10  
h["b"]=20  
puts h["a"]  
puts h["c"]
```

**Answer:**



**7. Prolog Execution (15 pts).**

- a. (6 pts total; 2 pts each.) Consider the following definitions. List all answers returned by the following queries (including substitutions of R that make the query true). If there is more than one answer, list them all; if there is no answer, write false. Answers must be listed in the correct order to receive full credit.

```
enrolled(steve,cm330).
enrolled(bill,cm351).
enrolled(alice,cm330).
isstudent(P) :- enrolled(P,_).
```

```
married(joe,alice).
married(bill,jane).
```

```
ismarried(P) :- married(P,_).
ismarried(P) :- married(_,P).
```

i. enrolled(R,cm330)

ii. isstudent(R).

iii. ismarried(steve).

.

- b. (3 pts) Suppose we extend the program above with the following definitions. What is returned by the query `unmarried_student(R)`?

```
hah(X) :- X, !, fail.
hah(_).
```

```
unmarried_student(X) :-
    isstudent(X),
    hah(ismarried(X)).
```

- c. (3 pts) What if we changed `unmarried_student` to be defined as follows; would that change the result of the above `unmarried_student(R)` query? Why or why not?

```
unmarried_student(X) :-  
    hah(ismarried(X)),  
    isstudent(X).
```

- d. (3 pts) Consider the following definitions. If `check(L)` is true, what can we say about the contents of `L` (explain in words)?

```
splast([X],[],X).  
splast([H|T],[H|T1],X1) :-  
    splast(T,T1,X1).
```

```
check([]).  
check([X]).  
check([H|T]) :-  
    splast(T,T1,H),  
    check(T1).
```

**8. Prolog Programming (10 pts).** Do *either* the next two problems. **If you do both, we'll average the two scores.**

- a. Define predicate `drop(L,X,P)` which holds when `P` is the same as the list `L` but with all occurrences of `X` in `L` removed. Thus, the following should hold:

```
drop([1,2],1,[2]).  
drop([1,1,2],1,[2]).  
drop([2,1,1],5,[2,1,1]).
```

- b. Define predicate `exp(N,M,P)` which holds when `P` is  $N^M$ . (You may assume that `M` is non-negative.) Thus, the following should hold:

```
exp(2,3,8).      % 23 = 8  
exp(3,2,9).      % 32 = 9 etc.
```

9. **Multithreading in Ruby (18 points).**

- a. (8 pts) Write all possible outputs from executing the following multithreaded Ruby program.

```
def func1
  i=2
  while i<=3
    sleep(rand())
    puts "func1: #{i}"
    i=i+1
  end
end

def func2
  j=0
  while j<=1
    sleep(rand())
    puts "func2: #{j}"
    j=j+1
  end
end

puts "Start"
t1=Thread.new{func1()}
t2=Thread.new{func2()}
t1.join
t2.join
puts "End"
```

- b. (10 pts.) The following code, implementing a thread-shared, fixed-sized Stack, has two problems: it has data races and it uses busy-waiting. Fix the code by using proper synchronization and condition variables. Write your changes to the side, clearly marking what you are inserting/deleting/changing. You may use the library functions listed here:

```
m = Monitor.new           // creates a new monitor
m.synchronize { ... }    // acquires m, enters the block, releases m
c = m.new_cond           // returns new condition variable (condvar)
for m
  c.wait_while { ... }    // waits on condvar c while code block is true
  c.wait_until { ... }    // waits on condvar c until code block is true
c.broadcast               // signals all threads waiting on condvar c
```

```
class Stack
  def initialize(sz)
    @idx = 0
    @buf = Array.new(sz)
  end

  def push(x)
    while @idx >= (@buf.size - 1) do
      # wait
    end
    @buf[@idx] = x
    @idx = @idx + 1
  end

  def pop
    while @idx == 0 do
      # wait
    end
    @idx = @idx - 1
    x = @buf[@idx]
    @buf[@idx] = nil
    x
  end
end
```

Example: s = Stack.new(2)  
t1 = Thread.new { s.push("hi"); x = s.pop; puts x }  
t2 = Thread.new { s.push("there"); y = s.pop; puts y }  
t1.join; t2.join

# should print hi\n there\n or there\n hi\n

10. Regular expressions, Context-free Grammars, Automata (12 pts).

a. (3 pts) Write a context-free grammar that accepts all strings  $a^n b^m$  where  $n = 2+2m$ , for all  $m > 0$ . For example, the strings aaaab, aaaaaabb, and aaaaaaaabbb are all in the language.

b. (3 pts) Give a regexp that denotes the language of all strings over  $\{a,b\}$  which start with a, end with b, and contain the substring abb.

c. Circle all of the inputs accepted by the NFA below (1 pt each)

- i.  $\epsilon$
- ii. aba
- iii. aaaaaaa
- iv. abab
- v. aaaab
- vi. aa

