CMSC330 Fall 2015 Final Exam

Name: _____ **UID:** _____

Discussion Time:	**10am**	**11am**	**12pm**	**1pm**	**2pm**	**3pm**
TA Name (Circle):	**Adam**	**Maria**	**Chris**	**Chris**	**Michael**	**Candice**
	Amelia	**Amelia**	**Samuel**	**Josh**	**Max**	

Instructions

- The exam has 15 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	/16
2	Lambda Calculus	/8
3	OCaml Types	/8
4	OCaml Execution	/20
5	OCaml Programming	/12
6	Ruby Programming	/15
7	Prolog Execution	/12
8	Prolog Programming	/12
9	Regexps, FAs, CFGs	/12
10	Security	/15
	TOTAL	/130

1. Concepts (16 points)

(a) (2 pts) True or **false**: A closure must be created every time a function is called.

(b) (2 pts) True or **false**: When using call-by-value parameter passing, arguments to functions are evaluated at the last possible moment, just before they're needed.

(c) (2 pts) **True** or false: In Java, if *S* is a subtype of *T*, then *S[]* is a subtype of *T[]*.

(d) (2 pts) List one benefit of using static type checking (1-2 sentences).

Detects errors at compile time

(e) (2 pts) List one benefit of using dynamic type checking (1-2 sentences).
Catches errors in runtime, Greater flexibility in programming

(f) (2 pts) Fill in the blanks below. Possible choices: *implicit*, *inferred*, *explicit*, *automatic*.

Java uses **explicit** declarations.

Ruby uses **implicit** declarations.

(g) (2 pts) Fill in the blanks below. Possible choices: *call-by-value*, *call-by-name*, *call-by-reference*, *call-by-push-value*.

Call by name is also known as lazy evaluation.

Call by value is also known as eager evaluation.

(h) (2pts) Circle the languages that use garbage collection.

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

2. Lambda calculus (8 points)

At the Jedi Academy, Wesley Crusher was talking to Darth Vader while preparing for their CS final. “I can’t believe it’s the year 40,000 BC and we still have to study this!”

Fully reduce the following lambda calculus terms

a. (3 pts) (λx.λy.λz.x y z) a b

λz. a b z

b. (3 pts) (λx.λy.λz.x y z) y z

λa. y z a

c. (2 pts) Indicate which (either or both) of the following pairs of terms are alpha equivalent.

i. (λx.x) y (λx.x) z **NO**

ii. (λx.λy.x y) a b (λa.λb.a b) a b **YES**

3. OCaml Types (8 points)

a) (2 pts) What is the type of the following OCaml expression?

```
[([2], “Force”); ([3], “Awakens”); ([1], “The”)]
```
(int list * string) list

b). (2 pts) What is the type of the following OCaml expression?

```
fun darth -> fun vader -> 2+vader
```
‘a->int->int

c) (2 pts) What is the type of the following OCaml function?

```
let han solo = (solo 1)
```
(int-‘a)->’a

d. (2 pts) Write an OCaml expression with type `int -> int list list`

fun x->[[x+1]];;

4. OCaml Execution (20 points)

Consider the following code statements in **bold**. If the statement doesn't compile; say that. If it runs, indicate what it will do; i.e., if it returns normally, indicate what value it will return; if it throws an exception, indicate that; if it runs forever, say that; if it prints something (also) say what it is.
Note: When there are multiple statements (separated by ;;) and one fails (due to exception, infinite loop, or type error), consider the statements that follow it as if the first had *not* failed.
Note: In OCaml, integer division rounds down. So `3 / 2 = 1` and `1 / 2 = 0`.

a) (6 pts)

```
let rec heal hp s =
 if hp = 0 then failwith "dead!"
 else if s = 0 then hp
 else heal (hp + s/2) (s-1)
;;
heal 5 0;;            5

heal 3 3;;            5

heal 0 3;;            Exception: Failure "dead!"
```

b) (4 pts)

```
type move =
 | Saber_thrust
 | Parry
 | Saber_slice
 | Jump_sideways

let damage x y =
 match x with
   | Saber_thrust -> if y = Jump_sideways then -1 else 1
   | Saber_slice -> if y = Parry then -1 else 1
   | _ -> 0
;;
damage Saber_thrust Saber_thrust;;          1

damage Parry Saber_slice;;                  0
```

d) (4 pts)

```
let make_move m1 m2 =
 let m1_damage = damage m1 m2 in (* damage was defined above *)
 let m2_damage = damage m2 m1 in
 (m1_damage,m2_damage)
;;
make_move Saber_thrust Saber_thrust;;        (1, 1)

make_move Parry Saber_slice;;                (0, –1)
```

c) (6 pts)

```
let rec blip ls ms =
 match (ls,ms) with
     ([],[]) -> []
   | (h1::t1,h2::t2) -> (h1,h2)::(blip t1 t2)
;;
blip [1;2] ["3";4];;

blip ["hello"] ["hello"; "bye"];;

blip [Parry; Saber_slice] [-1; -2];;        [(Parry, –1); (Saber_slice, –2)]
```

5. OCaml Programming (12 points)

Implement **any two of the next three functions in OCaml**. If you implement all three, we will grade them all (6 points each) and scale the final result by ⅔. **For parts (a) and (c), you must use the function `fold` in your implementation**, given on the next page (equivalent to OCaml's `List.fold_left`). If you do not use it, you will receive at most half credit for those problems.

a) **Implement the function `fight`** of type **`(move*move) list -> int*int`.** The function should apply the `make_move`, given above, to each pair of elements in the input list, summing all of the results, pairwise. (You should use `fold` in your implementation.) Examples:

```
fight [] = (0,0)
fight [(Saber_thrust,Saber_thrust)] = (1,1)
fight [(Parry,Saber_thrust);(Saber_slice,Saber_slice)] = (1,2)
```

```
let fight ms =
    let f (x1,y1) (x2,y2)=
        left d1,d2 = make_move (x2,y2) in (x1+d1,y1+d2) in
        fold f (0,0) ms
```

b) **Write a function `alternateList` of type `'a list -> 'a list`** which takes in a list `lst` and returns a list containing every other element of the original list. (You can use `fold`, or not.)

```
alternateList [] = []
alternateList [1;2;3;4;5] = [1;3;5]
```

```
let alterlist lst  x =
    let rec alist x n =
    match x with
        [ ]->[ ]
        |h::t->if x mod 2 = 0 then alist t (n+1) else h::(alist t (n+1)) in
    alist x 1
```

OR

```
let rec alternatelist l = match l with
    [ ]->[ ]
    [h]->[h]
    |h::h2::t->h::alist t
```

c) **Write a function `thresholdMin` of type `int -> int list -> int`**. Given an integer x and a list of ints `lst`, returns the smallest element of the list which is *strictly* greater than the x. For an empty list or if no elements are greater than *x*, return 0. (Recall you should use `fold` in your implementation.)

```
thresholdMin 3 [4;2;3;6] = 4
thresholdMin -2 [] = 0
thresholdMin 4 [1;2;3] = 0
```

```
let thresholdMin2 x lst = match lst with []->0| h1::t1 as ls->
                                List.fold_left ( fun a h ->
                                        if (a = 0 && h > x) then h
                                        els if(h > x && h < a) then h
                                             else a
                                 ) 0 ls
;;
```

Here is `fold`, for your reference:

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

6. Ruby Programming (15 points)

(a) (6 pts) Welcome to the rebel alliance! We heard you are a computer whiz so we are asking for your help with cracking a code. We have intercepted the Empire's transmissions and we believe we have found out how to decrypt them. In order to find the true message hidden in the transmission, we look for lines that

start with the letter `e`,
followed by a *number*,
followed by a space,
and then a list of words separated by spaces.

The number is the index (starting at 1) of a word in the list that is part of the secret message.

We've written a script to decrypt the message, but we need your help to fix it. Given an array of lines from the encrypted message, the function **`decrypt`** should generate an array of words that constitute the decrypted message. For example: if the input is a single line **e2 R2-D2 C-3PO Chewbacca Maul,** then the hidden message is **[C-3PO].**

We've tried to write the `decrypt` function below, but it doesn't work. In particular, when we run it on the above input, it fails on line 5 with the error *NoMethodError: undefined method `[]' for nil:NilClass*. **Fix the bugs so that it works.**

```
1  def decrypt lines
2    words = []
3    lines.each{|l|
4      if l =~ /^e\d+ .*/ then
5        words.insert(-1, $2[($1.to_i)-1])
6      end
7    }
8    words
9 end
```

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.

(b) (3 pts)

```
s = “123a456”
if s =~ /^(\d+)(\d+)$/
  puts $1
  puts $2
else
  puts “NOT Found”
end
```

answer: NOT Found

(c) (3 pts)

```
h = {}
h[0] = {}
h[0][0] = 0
puts h.has_key? 0
puts h.has_value? 0
```

answer:
TRUE
FALSE

(d) (3 pts)

```
m = [[1,6],[2,5],[3,4]]
m.each { |lst| lst.map! { |x| -x }}.sort!
puts m
```

answer:
–3
–4
–2
–5
–1
–6

7. Prolog Execution (12 points)

For this problem, consider the following definitions.

```
sith(plagueis).
jedi(yoda).

mentor(plagueis,sidious).
mentor(sidious,maul).
mentor(yoda,dooku).
mentor(dooku,qui_gon).
mentor(qui_gon,obi_wan).
mentor(obi_wan,anakin).

sith(X) :- mentor(Y,X), sith(Y),!.
jedi(X) :- mentor(Y,X), jedi(Y).

fights(X,Y) :- jedi(X),sith(Y).
fights2(X,Y) :- jedi(X),!,sith(Y).
```

For each query, list the substitution X that Prolog produces that makes the query true. If there is more than one answer, list them all, separated by semi-colons. If there is no answer, write false. Answers must be listed in the correct order to receive full credit.

a) ?- `sith(X).`
plagueis;
sidious

b) ?- `fights2(X,sidious).`
yoda

c) ?- `fights(anakin,X).`
plagueis;
sidious

d) ?- `fights(X,maul).`
yoda;
dooku;
qui_gon;
obi_wan;
anakin

8. Prolog Programming (12 points)

Implement **either of the following two functions**. If you implement both, we will grade both and average the two scores. You may write helper functions if you like. To receive full credit, each function should return at most one answer (e.g., with the help of cuts, if needed).

a) Implement `intval(L,R)`, where you are given `L`, a list of integers, and `R` is `L` in place-value interpretation, where the first element is the ones, the second is tens, the third is 100s, etc. Examples:

```
?- intval([1,2,3],R).
R = 321.
?- intval([],R).
R = 0.
?- intval([7,0],R) .
R = 7.
```

```
intval([],0).
intval([X],X) :- !.
intval([H|T],R) :-
  intval(T,S),
  S1 is S*10,
  R is H+S1.
```

b) Implement `max(L, M)`, where `L` is a list of integers, and `M` is the largest element in `L`. `max([],X)` is always `false`. Examples:

```
?- max([2,6,3,5],M).
M = 6.
?- max([],M)
false.
```

```
max([X], X) :- !.
max([H|T],H) :-
   max(T,N),
   H >= N.
max([H|T],N) :-
   max(T,N),
   N > H.
```

9. Regexps, FAs, CFGs, Parsing (12 points)

(a) (2 pts) Fill in the blank: The grammar below is not suitable for a recursive descent parser

because it is **_has overlapping first sets__and left recursive_____**

S - > acT | aaU | a| b
T - > Ta | b
U -> c

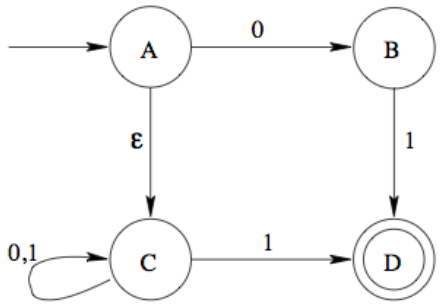
(b) (3 pts) Rewrite the grammar to avoid the problem. You are allowed to introduce new nonterminals.

S-aB|b
B->aT|aU|e
T->aT|b|e
U->c

(c) (3 pts) Write a regular expression that matches syntax corresponding to an integer array literal in Ruby. Assume there are *no empty lists* and all numbers have at least one digit. Some examples of strings accepted by your regex include [1], [1,2,3,4], and [11,872,90]

/^\[\d+(,\d+)*\]$/

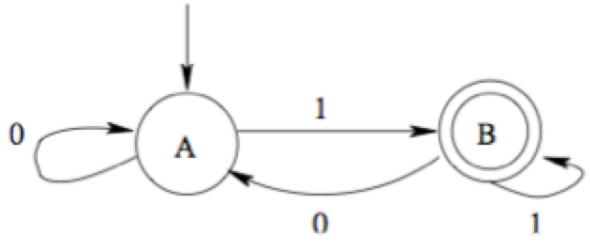
(d) (4 pts) Consider the following NFA, which recognizes set of strings over {0,1}



(2 pts) Give an equivalent regular expression

(1|0)*1

(2 pts) Give a DFA with two states that recognizes the same language. (Hint: DO NOT convert the given NFA to DFA. Design your DFA from the regular expression in above. Explicitly indicate your start state.)



10. Software Security (15 points)

a) (3 pts) If I have a buggy FTP server that allows user `josh` to illegally read the contents of user `saurav`'s files, then this is a violation of which property? (Circle the right choice)

Confidentiality Integrity Availability **Authorization** Authentication Audit

b) (3 pts) What do command injection, SQL injection, cross-site scripting, buffer overflows, and other security vulnerabilities have in common? (Choose the best answer.)

a. They leak private information to an adversary
b. The effect of the attack occurs on a network server
c. User input intended to be used as data is treated as code
d. Sanitization of user input is not a workable defense against them

c) (9 pts) Suppose we extended the FTP server from project 6 with a command *size* that outputs the size of a specified file. Below is code we might add to `ftp.rb` for this command; the input from the remote client is in the `arg` parameter (which should be the name of the file):

```
def size(path,socket)
 unless logged_in socket
   my_send(socket, "You're not logged in yet")
   return
 end
 if path =~ /^public\// then
   full_path = File.join(Dir.pwd,"users",path)
 else
   full_path = File.join(@s2a[socket].directory,path)
 end
 my_send(socket, "Size: \n#{`stat -f\"%z\" #{full_path}`}")
end
```

The code uses the `stat` command to compute the size of the file (per the %z specifier). This code is susceptible to **command injection**.

i) (3 pts) Mark where in the code above the problem is.

ii) (3 pts) Give an input that the client could use to exploit the vulnerability, i.e., to perform command injection.
.

size public/readme; rm –f /etc/password

iii) (3 pts) Show how to fix the code to eliminate the problem. If you can't remember particular Ruby commands or syntax, explain as best you can.
sanitize path