CMSC330 Fall 2016 Final Exam
Solution

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	/10
2	Lambda Calculus	/10
3	OCaml Types	/10
4	OCaml Execution	/12
5	OCaml Programming	/14
6	Ruby Programming	/16
7	Prolog Execution	/12
8	Prolog Programming	/8
9	Regexps, FAs, CFGs	/22
10	Operational Semantics	6
11	Security	/10
	TOTAL	/130

1. PL Concepts (10 pts)

A. (1 pts) (**T** / F) In call-by-name evaluation, arguments are evaluated just before they are needed. If an argument is not used in the function body, the argument is never evaluated.

B.

(1 pts) (T / **F**) The following code snippet uses tail recursion:

```
int factorial(int n) {
if (n == 0)
      return 1
else
      return n * factorial (n - 1);
}
```

C. (1 pts) (T / **F**) Consider the "is" operator in Prolog. When evaluating "LHS is RHS", both LHS and RHS are evaluated and match the corresponding results.

D. (1 pts) (**T** / F) If e1 and e2 are physically equal, then they are structurally equal.

E.
(2 pts) Consider the following OCaml code.
what is the value of f() using static and dynamic scoping?

```
let  a = 1 ;;
let f = fun () -> a + 10;;
let a = 2 ;;
f ();;
```

static: 11 **dynamic: 12**

F.

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

(3 pts) Consider the following C code.

Give the output if C uses call-by-value:	
Give the output if C uses call-by-reference:	
Give the output if C uses call-by-name:	

G. (1 pts) Which following term(s) is not a garbage collection (GC) technique?
a) mark & sweep
b) stop & copy
c) **malloc & free**
d) reference counting

2. Lambda Calculus (10 pts)

A. (1 pts) A free variable in lambda calculus is: (choose one answer only)
a) A variable in a lambda expression that is not given any arguments
b) A variable in an irreducible lambda expression
c) A variable that is not bound to any lambda abstraction
d) A variable that is bound to an outermost lambda abstraction

B. (2 pts) Underline the free variable(s) in this expression (λv.λw.**x** (w v) (λy. w) **y**) (λu.u **v**)

C. (3 pts) Make the parentheses explicit in the following expression

(x (λ x .**(((** x y**)** x**)** z**)**) **)** (λ w . **(** λ x . **(** x w **))**
)

(x (λx. (((x y) x) z))) (λw. (λx. (x w)))

D. (4 pts) Reduce the following lambda expression. Show all alpha-conversion and beta-reduction steps

(λx.λy.λz.y x z) y (λw.w w)

(λx.λa.λz.a x z) y (λw.w w)	Alpha
(λa.λz.a y z) (λw.w w)	Beta
λz.(λw.w w) y z	Beta
λz.y y z	Beta

3. OCaml Types (10 pts)

A. (4 pts) Write the types of the following expressions:

a) (fun e g -> e g g)
Solution: ('a -> 'a -> 'b) -> 'a -> 'b

b) let awake p = fold (fun a h -> a || h = "coffee") false p
Solution: string list -> bool

B. (6 pts) Provide expressions (without type annotations) that match the following types:

a) `a -> `b -> (`a -> `b -> `a) -> bool
Possible Solution: fun x y z -> (z x y) = x

b) (int * char) -> int -> bool
Possible Solution: fun (x,y) z -> x + z = 1 && y = ‘c’

4.OCaml Execution (12 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 <u>syntax</u> errors in these programs**).

A. (4 pts)

```
let result = let result = 7 in let result = result * 3 in result +
2;;
        result = 23
```

B. (4 pts)

```
        let rec f m = match m with
                [] -> 0
| h::h1::t when h = h1 -> 1 + (f t)
              | h::t -> 1 + (f t);;
```

```
      let result = f [“eggs”; “oj”; “toast”; “toast”; “bacon”;
“ham”];;

      Result = 5
```

C. (4 pts)

```
  let result =
   map (fun x-> 2* (fold(fun a h -> a + h) 0 x)) [[1;2;3];[2;3;4]];;

    Result = [12;18]
```

5.OCaml Programming (14 pts)

A. (8 pts) Write a function **mirror** of type **‘a list ->(‘a * int)-> ‘a list** that reverses each side of the list relative to a given separator, which is the tuple of an item and its index in the list. Assume index is in range, and the item at the index is correct. List has at least one item, the separator. You are allowed to use List.append and @, NOT List.rev or other functions in the List module.

Example:

mirror ["eggs"; "yogurt"; "oatmeal"; "pancakes"; **"cat"**; "oregano"] (“cat”,4)
Output: string list = [“pancakes”; “oatmeal”; “yogurt”; “eggs”; **“cat”**; “oregano”]

mirror ["eggs"; "bagel"; **"milk"**; "pancakes"; "cat"; "oregano"] (“milk”, 2)
Output: string list = ["bagel"; "eggs"; **"milk"**; "oregano"; "cat"; "pancakes"]

Didn’t spend a lot of time thinking about clever sol - if anyone has one feel free to post

```
let rec reverse l = match l with
[] -> []
| h::t-> (reverse t)@[h];;

let rec getleft l k = match l with
[] -> []
|h::t-> if k = 0 then [] else h::(getleft t (k-1));;

let rec getright l k = match l with
[]->[]
```

```
|h::t-> if k = 0 then t else getright t (k-1);;

let mirror l (x,y) = (reverse (getleft l (y)))@[x]@(reverse (getright l (y)));;
```

B. (6 pts) A multiway tree is composed of a root element and a (possibly empty) set of successors which are multiway trees themselves. A multiway tree is never empty. A multiway tree is defined as:

```
type 'a mult_tree = Node of 'a * 'a mult_tree list;;

let t1 = Node(10,[]);;

let t2 = Node(10,[Node(20,[Node(25,[])]); Node(30,[]); Node(40,[])]);;
```

Write a function countNodes that counts the nodes of a multiway tree.
For example:
countNodes t1 returns 1
countNodes t2 returns 5

Solution:

let rec countNodes (Node(v, l)) = List.fold_left (fun a b -> a+countNodes b) 1 l
;;

6.Ruby Programming (16 pts)

Write the output of the following Ruby programs. If the program produces an error of some sort, write the output up to that error and then write FAIL.

A. (3 pts)

```
ingredients = {"eggs" => 2, "bacon" => nil, "pepper" => 0}
ingredients["cheese"] = 6
t = []
ingredients.each{ |x, y|
  if y then
    t.push(x)
  end
}
puts t.sort
```

Answer (in order):

```
cheese
eggs
pepper
```

B. (2 pts)

```
j = [2, "egg", 4, "milk"]
j.each{ |x|
  puts j[0]
}
```

Answer (in order):

```
2
2
2
2
```

C. (11 pts)

Saurav Das runs a diner and he needs help keeping track of the orders. Below you must define the Ruby class Diner. In this class, we want you to define a FIFO queue for placing and cooking orders. Write the following methods for the Diner class:

initialize (3 pts)- The Diner class will always be created with no parameters, so use this method simply to set up your data structure(s)
place_order(order) – (5 pts)
This method adds a table's order to the FIFO queue. An order will be a single string with the following format:

"ITEM:COUNT,ITEM:COUNT,… ITEM:COUNT"

An example of this in practice would be:

"Omelette:2,Eggs:2,Pancakes:1"

There must be at least one item-count pair, and there is no limit to the number of pairs. Items can only contain alphabetical characters. Items on the order are added to the queue from left to right. And multiple times if its count is larger than 1. In the example above, the queue after adding the order would look like this:

Omelette, Omelette, Eggs, Eggs, Pancakes

A count of zero IS valid, while count is negative or a decimal is not valid. If an order has an invalid format, you should return nil, and not add anything to the queue. Return true if items are correctly added to the queue.

cook_order – (3 pts) This method dequeues the first item and returns it so Saurav can cook it

One possible solution:

```
class Diner
     def initialize
          @queue = []
     end

     def place_order (order)
          order = order.split(/,/)
          order.each { |x|
               if x ~= /(\w+):(\d*)/
                    $2.to_i.times { |y|
                         @queue.push($1)
                    }
               else
                    return nil
               end
          }
          return true
     end

     def cook_order
          @queue.shift
     end
end
```

7.Prolog Execution (12 pts)

For this problem, consider the following definitions.

```
ingredient(egg).
ingredient(oil).
ingredient(tomato).
ingredient(potato).
ingredient(lettuce).
beverage(milk).
fruit(strawberry).
fruit(banana).

food(potato,oil).
food(egg,tomato).
food(potato,lettuce).
food(egg,potato).
```

```
drink(X,Y) :- beverage(X),fruit(Y).
breakfast(X,Y) :- food(egg,X),drink(milk,Y).
breakfast(X,Y) :- ingredient(X),X \= Y,ingredient(Y).
breakfast2(X,Y) :- food(egg,X),!,drink(milk,Y).
breakfast2(X,Y) :-
ingredient(X),not(food(potato,X)),!,ingredient(Y),!.
```

For each query, list the substitutions 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. `(3 pts) ?- food(potato, X).`

```
X = oil ;
X = lettuce.
```

B. `(3 pts) ?- drink(milk, X).`

```
X = strawberry ;
X = banana.
```

C. `(3 pts) ?- breakfast(potato, X).`

```
X = strawberry ;
X = banana.
```

D. `(3 pts)  ?- breakfast2(X,Y).`

```
X = tomato,
Y = strawberry ;
X = tomato,
Y = banana.
```

8.Prolog Programming (8 pts)

Define a predicate **rotate(N,L,X)** which holds when X is the same as the list L is rotated N elements to the left. Thus, the following should hold:
rotate (2, [1,2,3,4,15], [3,4,15,1,2]).
rotate (0, [1,2,3,4,25], [1,2,3,4,25]).
rotate (7 [1,2,13,4,15], [13,4,15,1,2]).
You are not allowed to use any built-in predicates such as append.

```
rotate(_,[ ],[ ]).
rotate(_, [H], [H]).
rotate(N, L, X) :-
        length(L,N2),
        N3 is N mod   N2,
        length(Back, N3),
        append(Back, Front, L),
        append(Front, Back, X).
```

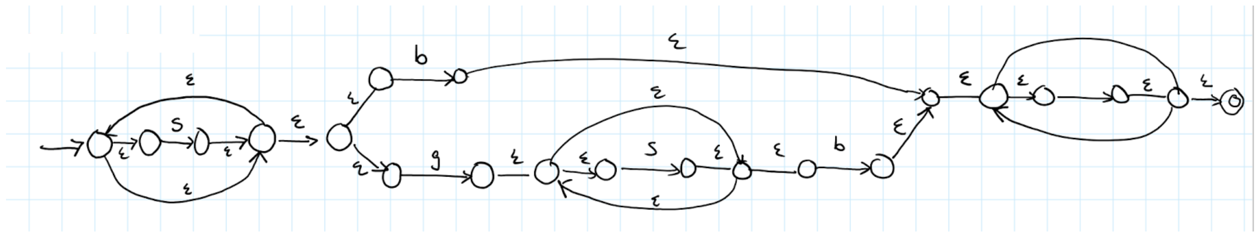
9.Regexps, FAs, CFGs (22 pts)

A. (3 pts) We're planning on opening a breakfast sandwich restaurant, but we want to make sure we haven't mistyped any of the prices. Write a Ruby regular expression to match only prices that have a dollar value of a single digit and two digits of cent values. Prices like $1.99 or $3.47 are acceptable, and $.49, $10.53, or 1.99 are NOT acceptable.

Key: ^\$\d\.\d\d$

B. (5 pts) We've written the following regex to match valid breakfast sandwich orders (combinations of egg, bacon, and spam). Convert it to a finite automaton:

s*(b|gs*b)s*



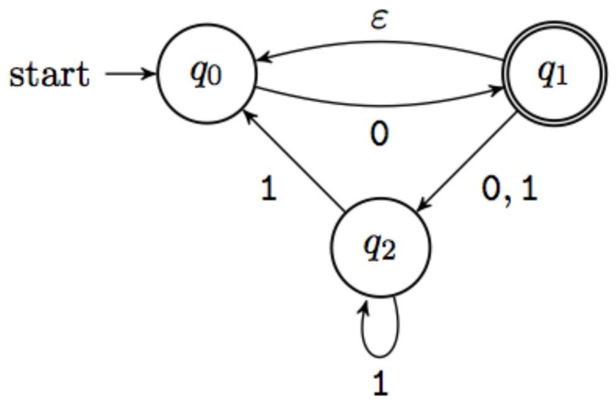
C. (5 pts) We wanted to validate coffee orders, too, but we can't write a recursive-descent parser for the grammar we wrote to match them. Identify the problem in our context-free grammar and fix it:

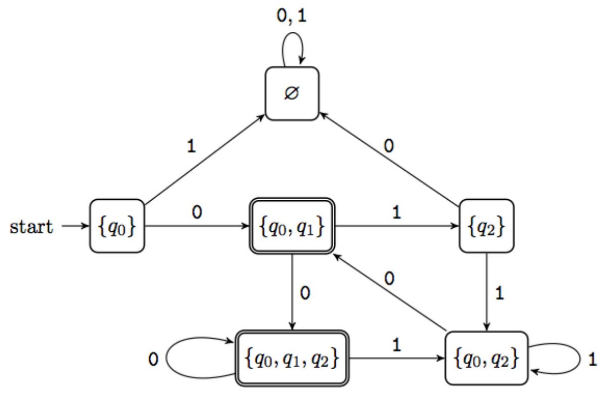
```
S -> small T | medium T | large T
T -> M latte | M cappuccino | coffee
M -> M soy | M half | M skim | epsilon
```

M is left-recursive

S -> small T | medium T | large T
T -> M latte | M cappuccino | coffee
M -> soy M | half M | skim M | epsilon

D. (5 pts) Convert the following NFA to a DFA using the subset construction method. The alphabet is Σ = {0,1}





E. (4 pts) Give a context-free grammar that generates the language
$L = \{a^{n+m} b^m c^n : m \geq 0,\ n \geq 0\}$

10. Operational Semantics (6 pts)

Given the following rules, write the natural deduction for the expression below.

*; let x = 5 in x + 3 ->[ ]

11. Security (10 pts)

A. (1 pts) If a language is dynamically typed, the compiler or interpreter prevents buffer overflows.

True. **False.**

B. (1 pts) Whitelisting is generally a more robust security technique than blacklisting.

True. False.

C. Consider the following code:

```
while true do
 print "Enter a directory to see what files are in it: "
 dirname = gets.chomp
 puts `ls -l #{dirname}`
end
```

(2 pts) Name the type of vulnerability in the code

Command injection

(2 pts) Describe or implement a technique to fix the vulnerability

1 pt for saying sanitize input, whitelist, blacklist

1 pt for saying that semicolon needs to be disallowed with one of the above techniques

OR

Implement a fix that prevents command injection and still allows normal functionality

D. (4 pts) Consider the following syntactically-correct code snippet from a Ruby webserver class implementing a menu service for a restaurant. The items in the menu are stored in a local MySQL database. Assume the MySQL class given can connect to and query this MySQL database properly via the new constructor and the query method respectively. For brevity, other web-related methods are not included in this snippet:

```
# imports, class definition and web server methods hidden
def initDB
    // connect to the local database named “menu”
    @db = Mysql.new('localhost', 'menu')
end

# getBreakfastItems is called immediately after breakfast items #are
queried by a user. maxCaloriesStr and maxPriceStr are raw strings
#inputted directly by the user

def getBreakfastItems(maxCaloriesStr, maxPriceStr)
      queryString = "select * from breakfast
                 where(#{maxCaloriesStr} and price <= #{maxPriceStr});"
      result = @db.query(queryString)
      if (result.error.nil?)
                  result.getResults()
             else
nil
      end
end
```

Indicate the biggest security risk present in the above code, and briefly state a potential solution.

1 pt: Notes SQL injection
1 pt: caused by lack of input sanitization/string interpolation
2 pt: Provides a valid potential solution using one or more of the following methods:
Prepared statements, blacklisting characters, whitelisting characters, some form of regex.