CMSC330 Spring 2018 Midterm 1
9:30am/ 11:00am/ 3:30pm

Name (PRINT YOUR NAME as it appears on gradescope):

__________________________________________________________________

Discussion Time (circle one) 10am 11am 12pm 1pm 2pm 3pm

Instructions
• Do not start this test until you are told to do so!
• You have 75 minutes to take this midterm.
• This exam has a total of 100 points, so allocate 45 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.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Programming Language Concepts</td>
<td>/10</td>
</tr>
<tr>
<td>2 Ruby Regular Expressions</td>
<td>/10</td>
</tr>
<tr>
<td>3 Ruby execution</td>
<td>/13</td>
</tr>
<tr>
<td>4 Ruby Programming</td>
<td>/18</td>
</tr>
<tr>
<td>5 OCaml Typing</td>
<td>/17</td>
</tr>
<tr>
<td>6 OCaml Execution</td>
<td>/15</td>
</tr>
<tr>
<td>7 OCaml Programming</td>
<td>/17</td>
</tr>
<tr>
<td>Total</td>
<td>/100</td>
</tr>
</tbody>
</table>
1. [10 pts] Programming Language Concepts

1.1 [7 pts] Circle the correct answer:

a. True / False: [1,2,3] is a list/array of three ints in both OCaml and Ruby

b. True / False: Static type checking occurs at compile time

c. True / False: In dynamically typed languages, a type error will go unnoticed if the line containing the error is never executed

d. The OCaml compiler does which of the following if you omit a case in a pattern match: Nothing / Emits a warning / Emits an error

e. True / False: Ruby variables are declared explicitly

f. True / False: All values in Ruby are objects

g. True / False: Ruby code blocks are first class, e.g., they can be stored in arrays

1.2 [3 pts] Show the contents of the closure for \( f \) after executing the following code:
\[
\text{let add} = (\text{fun } x \to (\text{fun } y \to x + y + 10));
\text{let } f = \text{add } 5;;
\]

<table>
<thead>
<tr>
<th>Code</th>
<th>Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2. **[10 pts] Ruby Regular Expressions**

2.1. **[3 pts]** Write a regular expression that accepts precisely 8, 9, or 10 letters

2.2. **[3 pts]** Write a string that matches the following regular expression:

```
/^www(\.[a-zA-Z]+)*(\.[a-zA-Z]{2,3})$/
```

2.3. **[4 pts]** Circle all of the given strings that match the following regular expression

```
/^\[0-9\]+\,(\[0-9\])\*$/
```

"3562"  "0432,7,7384"  "8392,6,3"  "8265,"
3.  [13 pts] Ruby execution

Write the output of the following Ruby code. If there is an error, then write ERROR. If nil is printed write “nil” and not the empty string. Hint: select invokes the block passing in successive elements, returning an array containing those elements for which the block returns a true value.

3.1.  [2 pts] Output:

```ruby
x = []
x[3] = 4
puts x["3"]
```

3.2.  [2 pts] Output:

```ruby
m = {"hello" => 3, "world" => 4}
puts m[3]
puts m["hello"]
```

3.3.  [2 pts] Output:

```ruby
x = {}
x["hi"].push(3)
puts x["hi"]
```

3.4.  [2 pts] Output:

```ruby
x = [2, false, 4, nil, 6, 0, 8]
puts x.select{|y| y}
```
3.5.  [2 pts]  
```ruby
x = "hello"
y = "hello"
puts (x == y)
puts (x.equal? y)
```

Output:

3.6.  [3 pts]  
```ruby
class Foo
  @@x = []
def initialize(ele)
    @@x.push ele
  end

def add(ele)
    @@x.push ele
    @@x
  end
end

f = Foo.new 5
g = Foo.new "hi"
puts (f.add true)
```

Output:
4. **[18 pts] Ruby Programming**

Implement a Graph class, which represents a *directed graph* as a collection of nodes that are linked by edges. *Cycles, including self-edges, are allowed,* but there can be *at most one edge between any pair of nodes.* A template for your implementation is given on the next page. You may **NOT** edit the initialize method, whose implementation implies you should store your graph as a hash. Implement the following methods.

4.1 **[8 pts]** `addEdge(str)` adds an edge represented by the `str` input parameter to the graph. The `str` input parameter has the format 'start: nodename end:nodename', where a valid nodename is a combination of one or more letters (uppercase or lowercase) followed by a dash ('-') followed by one or more digits. For example:

```ruby
g = Graph.new
g.addEdge("start: Node-5 end: tidepod-6")
g.addEdge("start: tidepod-6 end: A-7")
g.addEdge("start: A-8 end: tidepod-6")
```

will create a graph `g` with the edges (Node-5, tidepod-6), (tidepod-6, A-7), and (A-8, tidepod-6) in it. If the input string to `addEdge` is incorrectly formatted, then nothing will be added. For example:

```ruby
g.addEdge("start: Node5 end: hello-6")
```

will add no edges to `g` because Node5 is an invalid nodename.

4.2 **[5 pts]** `inDegree(node)` takes a node (a string) and returns the number of edges ending at that node. For example, for the graph `g` above, `g.inDegree("Node-5")` is 0, while `g.inDegree("tidepod-6")` is 2. The `inDegree` of a node with no incoming edges (or any edges at all) in the graph is 0.

4.3 **[5 pts]** `outDegree(node)` takes a node (a string) and returns the number of edges that start at that node. For example, for graph `g` above, `g.outDegree("Node-5")` and `g.outDegree("A-8")` are both 1. A node with no outgoing edges has degree zero, as does a node with no edges at all.

Implement your solutions on the next page.
class Graph
    def initialize  # do not change, add to, or delete this method
        @g = { }
    end

    def addEdge(str)

    end

    def inDegree(node)

    end

    def outDegree(node)

    end
end
5. [17 pts] OCaml Typing
Determine the type of the following definitions. Write ERROR if there is a type error.

5.1. [2 pts]

```ocaml
type 'a option = Some of 'a | None
let f a =
  if a < 0 then None else Some a
;;
```

5.2. [3 pts]

```ocaml
let f x y = [x;y]
;;
```

5.3. [3 pts]

```ocaml
let rec g l =
  match l with
  | [] -> []
  | [x] -> [x]
  | h1::h2::t -> (h1,h2)::(g t)
;;
```
Write an expression that has the following type, without using type annotations

5.4  [3 pts]  bool -> bool -> bool list

5.5  [3 pts]  (int * 'a) -> int

5.6  [3 pts]

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

    fold: ('a -> 'b -> 'a) -> 'a -> 'b list -> 'a

Define a function \( f \) that when used in the following expression will not produce any type errors. The implementation and type of \( \text{fold} \) are given for reference, above.

    fold f ([],0) [5;4;3;2;1]
6. [15 pts] OCaml Execution

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

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

Determine the final value of the following expressions. Write **EXCEPTION** if an exception is thrown or **ERROR** if there is a type error.

6.1. [2 pts]  
let f a =  
  if a = 1 then "harambe"  
  else 0 in  
    f 5

6.2. [3 pts] (you might find it useful to refer to the map and fold definitions given above)  
let xs = map (fun (x,y) -> x) [(2,"a");(3,"b")] in  
fold (fun a h -> a * h) 1 xs

6.3. [2 pts]  
let f a = fun b -> if a > b then a else b in  
map (f 1) [0;1;2;3]
6.4. [2 pts] let f a b = if a=b then (a-1) else (b+1) in
    f (4,8)

6.5. [3 pts] let y = 4 in
    let sub x y = x - y in
    let part = sub 3 in
    let y = 2 in
    (sub 3 7, part y)

6.6. [3 pts] (you might find it useful to refer to the type 'a option given in 5.1)
    let rec f l =
        match l with
        | [] -> 0
        | None::t -> f t
        | (Some _):t -> 1 + (f t)
    in f [Some "a"; None; None; Some "b"; Some "c"]
7.  [17 pts] OCaml Programming

7.1. [8 pts] Write a function `int_of_digits` that takes a list of digits and returns an int having those digits. **For full credit, you must implement `int_of_digits` using `fold`** (see the top of question 6 for its definition). Examples:

```ocaml
int_of_digits [] = 0
int_of_digits [0] = 0
int_of_digits [1;2;3] = 123
int_of_digits [1;0] = 10
```

Answer:

```ocaml
let int_of_digits lst =
```
7.2. [9 points] Using the int_tree type below, write a function sum_level that sums all the node values at a given level within the tree (starting at 0 for the top). Leaves present at a given level do not contribute (i.e., they have count zero). If the level is greater than the depth of the tree, return 0.

type int_tree =
    IntLeaf
  | IntNode of int * int_tree * int_tree

Examples:

sum_level (IntLeaf) 0 = 0;;
sum_level (IntLeaf) 1 = 0;;
sum_level (IntNode (1,IntNode(2,IntLeaf,IntLeaf),IntLeaf)) 0 = 1;;
sum_level (IntNode (1,IntNode(2,IntLeaf,IntLeaf),IntLeaf)) 1 = 2;;
sum_level (IntNode (1,IntNode(2,IntLeaf,IntLeaf),IntNode(3,IntLeaf,IntLeaf))) 1 = 5;;

Write your code here (add the rec keyword if you need it):

let sum_level t n =