CMSC 330 Exam 1 Fall 2021 Solutions

Q1. Introduction

…

Q2. PL Concepts

Q2.1. OCaml is statically typed while Ruby is dynamically typed. [✓ True]/False

Q2.2. Fill in the blanks such that the below expression demonstrates shadowing and returns 6.

```
let x = 5 in
let x = x + 1 in
x
```

Q2.3. Tuples in OCaml are homogeneous (same type) while lists are heterogeneous (can be formed by different types). True/[✓ False]

Q2.4. What is wrong with the following code?

```
let f x y = if x + 1 = y then x +. y else x;;
```

- Syntax error
- [✓ Type error]
- Both Syntax and Type error

Q2.5. Which of the following are objects in Ruby?

- [✓ `{1 => 2}`]
- [✓ `nil`]
- [✓ `[1, 2, 3]`]
- `{ |x| x + 1}`

Q2.6. What variables must be in the environment of the closure for function `foo` in the following code?

```
let foo =
  let c = ref 0 in
  let m = 2 in
  fun x -> c := !c + x * m;  !c
```

- `c`, `m`, and `x`
- [✓ `c` and `m`]
- `x`
- `m` and `x`
- `c` and `x`

Q2.7.

```
let next =
  let c = ref 0 in
  fun () -> c:=!c+1; !c
```

Select one of the following inputs so that the code below evaluates to `[1; 2; 3; 4]`

`List.map next` ________________

- `[1; 1; 1; 1]`
- `[1; 2; 3; 4]`
- **`[(); (); (); ()]`**
- `[0; 0; 0; 0]`

Q3. Regular Expressions

Q3.1. Write a regular expression to match on ISBN-13 numbers, ISBN-13 numbers have 13 digits and are of the format: `ISBN-13: <A>-<B>-<C>-<D>-<E>`

A: Prefix

B: Group identifier

C: Publisher identifier

D: Title identifier

E: Check digit

ISBN prefixes are 3 digits long, group identifiers are 1 digit, publisher identifiers are 2 or 3 digits long, title identifiers are 5 or 6 digits long, and check digits are a singular digit.

For example:

```
ISBN-13: 978-3-16-148410-0
ISBN-13: 978-1-876-86197-9
```

`^ISBN-13: \d{3}-\d{1}-\d{2,3}-(\d{5,6})-\d{1}$`

Q3.2. Make only one change to the following regular expression such that it exactly matches strings with the format: `MM/DD/YYYY`

```
^\d{1,2}\/\d{1,2}\/\d+$
```

`^\d{1,2}\/\d{1,2}\/\d{4}$`

Note: "One change" means that you can change more than one character in the regex, but only one point of functionality can change. Both of the following should be accepted: 5/2/1988 and 55/23/9999.

Q4. Ruby: Fill in the Blanks, Output, or Input

Q4.1.

```
h = {}
for i in 1..3 do
  if i % 2 == 0 then
    h[“even”] += 1
  else
    h[“odd”] += 1
  end
end
puts h[“even”]
puts h[“odd”]
```

What is the output of the above code? If it throws any error, type in "Error".

Error

Q4.2 Fill in the blanks so that the content of x is `["one", "two", "three"]`.

```
h = {1 => "one", 2 => "two", 3 => "three"}

x = h.keys.collect { |k| h[k] }
```

Q4.3. Fill in the blanks so that the following is printed:

```
class A method 1
module M method 1

class A
  def m1()
      puts "class A method 1"
  end
  def m2()
       puts "class A method 2"
  end
end

module M
   def m1()
      puts "module M method 1"
   end
end

class B < A
    include M
end

x = A.new
y = B.new

x.m1
y.m1
```

Q4.4. The function `expand` takes an array of two-element arrays where each tuple array contains a frequency `f` and an element x and returns an array of arrays such that each inner array contains `f` copies of x.

For example:

```
expand []  ==> []
expand [[1, '2'], [3, '4']]  ==> [['2'], ['4', '4', '4']]
```

Fill in the blanks to complete this implementation. (*Hint:* `Array.new(4){1} ==> [1, 1, 1, 1]`)

```
def expand(l)
  l.map { |freq, elem| Array.new(freq) { elem } }
end
```

Q5. Ruby: Coding

As students come back to campus since the COVID-19 outbreak, QR codes have been placed all over several buildings to enable contact tracing. You are given the task to write a program that reads contact tracing data to find the students who may have come into close contact with other students who test positive for COVID19.

You will be given a file that will include the data for the QR code scans. Each line of the file corresponds to a single scan that a student has made, and has the following format:

```
<firstname> <lastname>,<location>
```

You can assume that `firstname` and `lastname` will always be a student's first and last name, defined as a single upper-case letter, followed by at least one lowercase letter, and that `location` will always be three upper-case letters followed by 4 digits. You can also assume that there will be no duplicate lines, and that no student's name will appear more than once.

A short example of one of these files may look like the following:

```
David Smith,IRB0324
Michael Yang,IRB0324
Roger Eastman,IRB0324
John Chadley,ESJ0224
Master Yoda,VMH0201
Little Timmy,TWS1212
Covid Man,IRB0324
```

Your task is to fill in the blanks to complete the following class:

```
class covid_detector
    def initialize
        # Q1 TODO: Set up any data structures you may need
        @locs_to_students = Hash.new(nil)
        @students_to_locs = Hash.new
    end

    def read_files(scans_file)
        File.readlines(scans_file).each do |line|
            # Q2 TODO: Implement the body of this loop
            if str =~ /^([a-zA-Z]+\s[a-zA-Z]+),([A-Z]{3}\d{4})$/
                if @locs_to_students[$2] == nil
                    @locs_to_students[$2] = []
                end
                @locs_to_students[$2].push($1)
                @students_to_locs[($1)] $2
            end
        end
    end

    def close_contact(name)
        #Q3 TODO: Implement this function
        @locs_to_students[@students_to_locs[name]].select { |student| student != name }
    end
end
```

Q6. OCaml: Typing

Q6.1. Without using type annotations, write an OCaml expression that has type `int * int -> bool`

```
fun (x, y) -> x + y = 1
```

Q6.2. Without using type annotations, write an OCaml expression that has type `int list -> int -> float list`

```
fun lst x -> match lst with [] -> [float_of_int x] | h::_ -> [float_of_int h]
```

Q6.3. Without using type annotations, write an OCaml expression to fill in the blank so that entire expression has type `int -> int list`

```
(fun x -> (fun y -> [x; y])) 1
```

Q7. OCaml: Where's the Bug?

Identify what specific portion of the below code is causing the type error and what you can change to have it output the correct value.

```
let rec f a b = match a with
| [] -> []
| (x, _)::t -> (x, b) @ (f t b);;
```

For example:

```
f [(0, 0); (0, 0); (0, 0)] 1 = [(0, 1); (0, 1); (0, 1)]
f [(“h”, 3); (“I”, 15); (“j”, -3)] “d” = [(“h”, d); (“I”, d); (“j”, d)]
f [(1, “KIM”); (2, “AVW”)] “Iribe” = [(1, “Iribe”); (2, “Iribe”)]
```

The “@” operator above attempts to combine two lists, but the left part is a tuple, not a list. Fix this by replacing “@” with a cons operator (::) so that it appends a value to an existing list.

Q8. OCaml: What's the Input?

Q8.1.

```
let f a b c = a + b - c in
let g = f 12 in
let h = g 3 in
let a = 4 in
h x
```

What should x be for the expression to evaluate to 8? 7

Q8.2.

```
let op f = List.fold_left f 0 [1; 2; 3; 4; 5]
```

What should `f` be for `op f` to evaluate to 5?

`(max)` OR `(fun x y -> (x + y + 1)/2)` OR `(fun x _ -> x + 1)` OR `(fun _ x -> x)`

Q9. OCaml: Fill in the Blank

Q9.1. What should x be for `f x "a"` to evaluate to 3?

```
type 'a l =
| Pair of ('a * 'a l)
| Empty

let rec f x m =
  match x with
  | Empty -> 0
  | Pair(a, b) ->
      if a = m then
        1 + (f b m)
      else
        (f b m)
```

```
Pair("a", Pair("a", Pair("a", Empty)))
```

Q9.2. Complete the function areas that, when given a list of shapes, returns their areas as a list.

For example:

```
areas [Circle 3; Rect (3,4); Square(9)] = [27; 12; 81]
areas [] = []
```

Suppose shapes are defined as:

```
type shape =
| Circle of int
| Rect of int * int
| Square of int;;
```

Note: To keep things simple, use pi = 3

```
let areas lst =
  let areas_helper s =
    match s with
    | Circle r -> 3 * r * r
    | Rect (l, b) -> l * b
    | Square l -> l * l in
  fold (fun a x -> a @ [areas_helper x]) [] lst
```

Q10. OCaml: Coding

Q10.1. Given 2 lists of the same length, implement `merge_lists`, which merges them into one list that alternates elements from the first and second lists.

```
merge_lists [1; 5; 2] [7; 4; 8] = [1; 7; 5; 4; 2; 8]
merge_lists [1; 0] [1; 5] = [1; 1; 0; 5]
merge_lists [] [] = []
```

```
let rec merge_lists lst1 lst2 =
  match lst1, lst2 with
  | [], [] -> []
  | h1::t1, h2::t2 -> h1::h2::(merge_lists t1 t2)
```

Q10.2. Write a method add_k_n_times that inserts into a list `lst`, an element `k` exactly `n` times at a given index `i`. If `i` is greater than the length of the list insert at the end.

The arguments are (in order): list, element to add, number of times to add it, and index at which to add it.

```
add_k_n_times [2; 3] 4 2 0 = [4; 4; 2; 3] (* adds `4` two times at the index `0` *)
add_k_n_times [1; 5; 7] 3 1 1 = [1; 3; 5; 7]
add_k_n_times ["bad"; "good"; "meh"] "neat" 3 2 =
     ["bad"; "good"; "neat"; "neat"; "neat"; "meh"]
```

```
let rec add_k_n_times lst ele num i =
  let rec helper a n =
    match n with
    | 0 -> []
    | _ -> a::(helper a (n-1))
  in
  match lst with
  | [] -> helper ele num
  | _::_ as l when i = 0 -> (helper ele num) @ l
  | h::t -> h::(add_k_n_times t ele num (i-1))
```