Q1. Honour Pledge

Q2. OCaml

Fill in the blanks in function `mys` such that the expressions below are `true`

```
let rec mys ls ms =
  match (ls,ms) with
    ([],[]) -> []
   |([],_)->[]
   |(_,[])->[]
  | __A___ -> ___B____

mys [1;2] [3;4] =  [(1, 3); (2, 4)];;
mys ["A"; "B"] [-1; -2] = [("A", -1); ("B", -2)]
```

A

```
(h1::t1, h2::t2)
```

B

```
(h1,h2) :: (mys t1 t2)
```

Q3. OCaml

Function `foo` prints "positive" if the input is a positive number, otherwise, it will NOT print any message. Fill in the blank to correct the following code.

```
let foo x = if x > 0 then
               print_string "positive"
             else _______

()
```

Q4. OCaml

```
type move =
  | Left
  | Right
  | Up
  | Down
```

```
let damage x y = match x with
  |A ->    B
  |C ->    D
  | _ -> 0
;;
```

Fill in A,B,C, and D such that the following expressions are `true`

```
damage Left Right = 1
damage Right Up = 0
damage Up Left = 1
```

(Note: There is more than one right answer)

`Up`

- ( ) A
- ( ) B
- (X) C
- ( ) D

`Left`

- (X) A
- ( ) B
- ( ) C
- ( ) D

`if y = Down then -1 else 1`

- ( ) A
- (X) B
- ( ) C
- ( ) D

`if y = Right then -1 else 1`

- ( ) A
- ( ) B
- ( ) C
- (X) D

Q5. Ruby

Fill in the blank in the following ruby code, such that it should print "aptain".

```
a = "Captain America"
if a =~ /([a-z]+)/ then
  puts _______
end
output: aptain
```

Soln: `$1`

Q6. Ruby

Fill in the blank such that this code print "Heart"

```
a= {}
a[___A_____] = ____B_____
a["Spade"]["Club"] = "Heart"
puts a["Spade"]["Club"]
```

A: `"Spade"`

B: `Hash.new()`

Q7. Lambda Calculus

1. Free variables

Find all free (unbound) variables in the following Lambda-expression.

$$(\lambda x.\lambda y.\lambda x.xz)\lambda a.b\lambda b.b$$

z and the b between `\a` and `\b`

2. Beta-reduction

Reduce the following Lambda-expression to Beta-normal form. Be sure to show each reduction step for full credit. If the expression reduces infinitely, write "infinite reduction".

$$(\lambda a.(\lambda b.bb))c(\lambda d.dd)$$

$$(\lambda a.(\lambda b.bb))c(\lambda d.dd)$$

$$(\lambda b.bb)(\lambda d.dd)$$

$$(\lambda d.dd)(\lambda d.dd)$$

infinite reduction

3. Lambda

Are the following two lambda expressions alpha equivalent?

```
(\x.x) y                          (\x.x) z
```

```
No, cannot alpha convert free variables.
```

Q8 OCaml Typing

Write an expression that has the following type (without using type annotations)

Q8.1 (3 points)

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

Solution

```
fun (a,b)  c ->  c a
```

Q8.2 (3 points)

Fill in the blank such that the types of f is (int -> 'a) -> 'a

```
let f g = _______________;;
```

Solution

```
let f g = (g 2)
```

Q8.3 (3 points)

Fill in the blank such that the type of the expression int -> int

```
(fun x -> (_______ in x+y));;
```

Solution

```
(fun x -> (let y = 2 in x + y));;
```

Q9 OCaml programming

For the following questions, you may use the `List` module and `map`, `fold_left` and `fold_right`.

You may NOT use any other module or any other functions from the list module. You also may NOT use any imperative OCaml. You may make any of the functions recursive, and you also may define helper functions.

Q9.1 Mess

a. Define a type to represent the set `mess`. A `mess` is either `Dust`, a `Spill` that contains a `string` that says what is spilled, or a `Big` mess that contains two `mess`es.

```
type mess =
|Dust
|Spill of string
|Big of mess * mess
```

b. Then use your type to write the function `clean`. `clean` should take a `mess` and replace all the `spills` with `dust`. If the mess has no `spills` it should remain unchanged. For example:

```
clean Big(Dust, Big(Spill "Water", Big(Spill "Milk", Big(Dust, Big(Spill "Juice", Dust)))))
```

should return

```
Big (Dust, Big (Dust, Big (Dust, Big (Dust, Big (Dust, Dust)))))
```

```
let rec clean m =
match m with
|Dust -> Dust
|Spill _ -> Dust
|Big (m1, m2) -> Big (clean m1, clean m2);;
```

Q9.2 Numbers

If we use `Zero` to represent `0`, and `Next x` to represent the next number of `x`, we can represent natural numbers `num` as

```
type num =
    |Zero
    |Next of num;;
```

a. Write the function `to_int: num -> int` that takes a `num` and returns the equivalent arithmetic value of type `int`. For example: `to_int (Next (Next Zero)` returns 2.

```
let rec to_int num =
match num with
|Zero -> 0
|Next n -> 1 + to_int n;;
```

b. Write the functions `even` and `odd`, both with type `num -> bool`, that take a `num` and returns whether it is even or odd.

```
let even n =
(to_int n) mod 2 = 0;;

let odd n =
(to_int n) mod 2 = 1;;
```

---

10. Ruby Programming

In a hypothetical scenario, let's say you are a grading TA for a class called CMSC333. The grade distribution for CMSC333 will be split equally between 1 project and 1 exam. Both the project and the exam will be graded out of 100 points. The grade distribution on the project will be split as follows, 45% public tests and 55% release tests. There are 9 public tests and 11 release tests. Failed tests will get 0 point. You need to calculate the final grade as

```
final grade = exam grade * 50% + project grade * 50%
```

The file you are processing has the following format:

```
Student_First_Name
Exam_Grade
Project1_Test_Name, Project1_Test_Result
```

For example (the ellipsis "..." is not part of the file, but indicates the continuation of lines):

```
alice
89
Ptest0, 6
Ptest1, 4
Ptest2, 0

...

Rtest9, 8
```

```
Rtest10, 0
bob
95
Ptest0, 0
Ptest1, 4
Ptest2, 5

...

Rtest9, 0
Rtest10, 6
```

The first line of the file will be the name, the second line will be exam grade the student received, an integer between 0-100. Every line after the first will contain data for the Project. Project test names will start with a captial "P" or "R" (denoting either public or release), followed by the word "test" and the test number (test numbers start at 0). Project test names and test results will be separated by a comma (no spaces). In a file, there will only be 9 lines of public test data and 11 lines of release test data as there are 9 public tests and 11 release tests for the project. After that, it will be the next student.

You may assume that each line in the file has no preceding or succeeding spaces, you may assume no preceding zeros for any of the numerical data and that the project test results will add up to a number between 0-100, and you may also assume that the file follows the described format above precisely with no duplicate data/lines.

1. `initialize` and `processData`

Implement `initialize` and `processData`. Initialize whatever instance variables you see fit to have in the `initialize` method. In the `processData` method, process the data and return an array of [three element array] where the first element is the name, and the second element is the final grade, and third element is a string (either capital "P" or capital "F") that represents whether or not the student passed the class. A passing grade in CMSC333 is a 60.0 and above.

An example return for this method would be

```
[["Alice", 92,"P"],["david",58,"F"]]
```

Here is the starter code for you, please copy this code and add your implementation to it.

```ruby
def initialize
     #your code here
end

def processData
    IO.foreach("grades.txt") do |line|
```

```
        #your code here
    end
end

def initialize
    @database = []
    @examGrades = {}
    @projectGrades = {}
    @name = ""
    @studentList = []
end

def processData
    IO.foreach("grades.txt") do |line|
        examRegex = line =~ /^(100|[0-9]{1,2})$/
        publicRegex = line =~ /^(Ptest[0-8]),([0-9])$/
        releaseRegex = line =~ /^(Rtest[0-9]|10),([0-9])$/
        nameRegex = line =~ /^([a-zA-Z]+)$/


        if nameRegex then
            @name = $1
            @studentList << $1
        elsif examRegex then
            @examGrades[@name] = $1.to_i
        elsif publicRegex || releaseRegex then
            @projectGrades[@name] = []
            @projectGrades[@name] << $2.to_i
        end
    end

    @studentList.each do |student|
        finalGrade = ( (@examGrades[student] * (.5)) + (@projectGrades[student].sum * (.5))

        if finalGrade >= 60.0 then
            @database << [student, finalGrade.to_i, "P"]
        else
            @database << [student, finalGrade.to_i, "F"]
        end
    end

    return @database
end
```

Rust

Q11.1 Ownership

Consider the following Code snippet

```
fn main(){
    let mut a = String::from("Memory");
    {
        let c = &a;
    }
        let d = a;
    let f = &d;
    let b = a; // <---- HERE
    let g = &b;
    //HERE
}
```

which variable has ownership of `"Memory"`? If an error is thrown state the error and briefly explain why the error occurred.

An error is thrown. `d` has ownership of `a` so when `b` tries to get the ownership a compiler error is thrown.

```
  |
2 |     let mut a = String::from("Memory");
  |         ----- move occurs because `a` has type `std::string::String`,
                          which does not implement the `Copy` trait
...
6 |         let d = a;
  |                 - value moved here
7 |     let f = &d;
8 |     let b = a;
  |             ^ value used here after move
```

Q11.2 Rust Debugging

For the following code, write the expected output.

If there is an error in the code, rewrite it so that it will not throw an error and will not change the behavior, and write the output of the fixed code. Array is a primitive data type in Rust.

```
fn main () {
    let ar = ["1", "2", "3"];
    let ar1 = ar;

    for i in 0..3 {
        println!("{}", ar[i]);
```

```
    }

    for i in 0..3 {
        println!("{}", ar1[i]);
    }


    }
```

- ( ) Error
- (X) No Error
- Reason (if error):

```
No Error
```

- Output after fix:

```
1
2
3
1
2
3
```

Q12 Formal Languages Typing

Q12.1 (3 points)

What is the epsilon closure of the starting state 1 in `epsilon.png`?

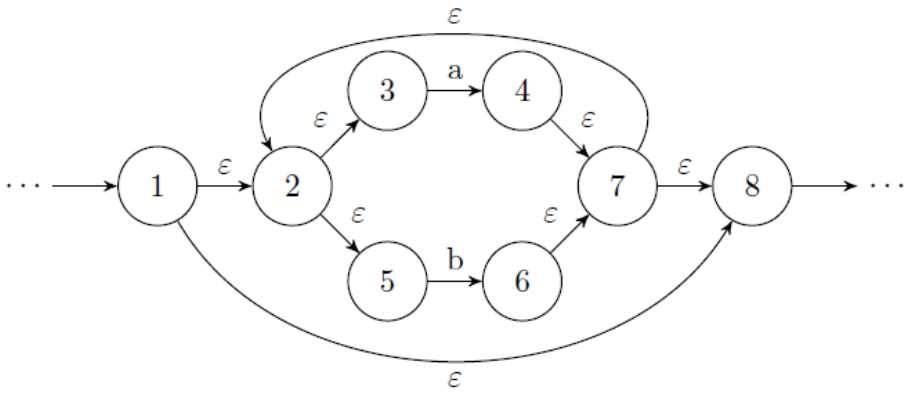

Figure 1: epsilon.png

Solution

```
{1, 2, 3, 5, 8}
```

Q12.2 (3 points)

Give an equivalent regular expression for the following NFA from `nfa.png`

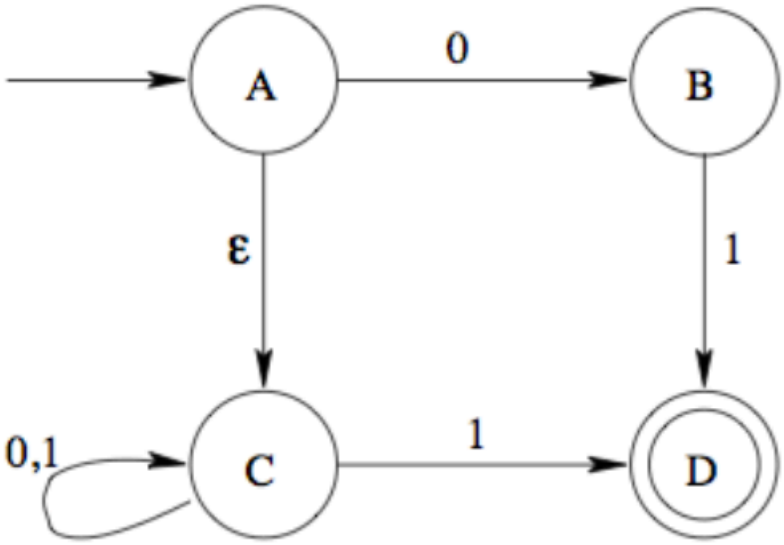

Figure 2: nfa.png

Solution

```
(X) 01|(0|1)*1
( ) 01|(10)*1
( ) ABD|(ACD)
( ) $$\epsilon$$
```

Q12.3 (3 points)

Write a CFG to accept any strings matching the following format

$$m^x b^y c^z$$

Where

$$y = (x + z),\ x > 0,\ z >= 0$$

Solution

$$S -> AB$$
$$A -> mAb|mb$$
$$B -> bBc|\epsilon$$

Q13 Parsing

Q13.1

For the following question refer to this grammar.

$$S \rightarrow S + S \mid T \mid aS$$

$$T \rightarrow bT \mid c \mid T - T \mid S$$

This CFG is ambiguous. Show that it is ambiguous by writing two different left most derivation for c-c-c Derivation A

```
S -> T -> T - T -> (T - T) - T -> c - T - T -> c - c - T -> c - c - c
```

Derivation B

```
S -> T -> T - T -> c - T -> c - T - T -> c - c - T -> c - c - c
```

Q13.2

For the following question refer to this grammar (n refers to the natural numbers).

$$\text{E} \rightarrow (\text{E}) \mid \text{E} + \text{E} \mid \text{if E then E else E} \mid \text{Val}$$

$$\text{Val} \rightarrow n \mid \text{true} \mid \text{false}$$

a. Write an AST for (1+2)+3 using the following AST of type `expr`

```
type expr =
  Int of int
 | Bool of bool
 | Sum of (expr * expr)
 | If of (expr * expr * expr)
```

```
(Sum (Sum (Int 1) (Int 2)) (Int 3))
```

b. Can a recursive descent parser parse this grammar?

( ) Yes (X) No

This grammar is ambiguous so a recursive descent parser can't parse it.

Q14 Semantics

Using the following rules(Opsem rules), indicate what should be written in place of each of the numbered text boxes in the following question.

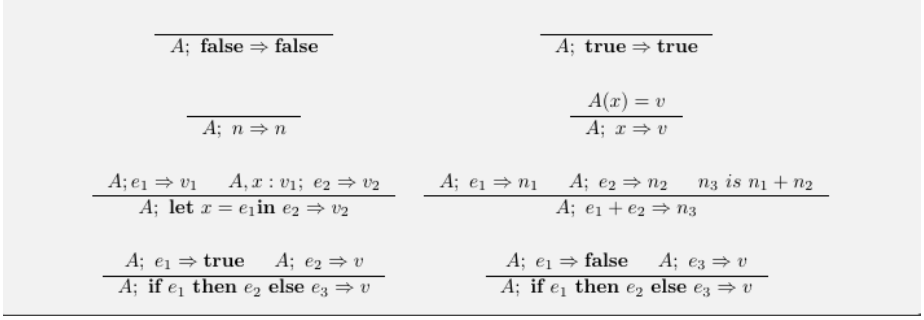

Figure 3: Opsem rules

1. Operational Semantics

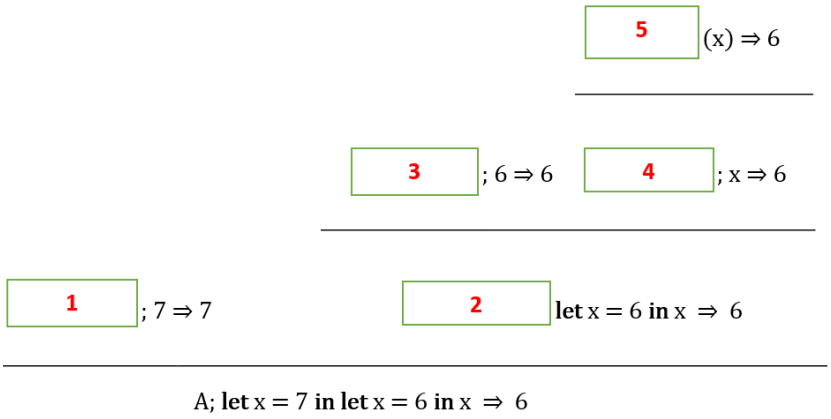

Figure 4: Opsem question

```
Box 1:  A
Box 2:  A, x = 7
```

```
Box 3:  A, x = 7
Box 4:  A, x = 7, x = 6
Box 5:  A, x = 7, x = 6
```

Q15 Bonus (4 Points)

How are you feeling?

Suggested Solution

```
The only accepted solution is I am feeling sad that
I will no longer be taking CMSC330 aka the best class!
```
