Exam 1

STUDENT NAME

Search students by name or email...

Q1
0 Points

Please carefully read the instructions below:

Ground Rules

This exam is open-note, which means that you may refer to your own notes and class resources during the exam. You can also use `irb` and `utop` (or other programs). You may not work in collaboration with anyone else, regardless of whether they are a student in this class or not. If you need to ask a question about the exam, post a private question on Piazza.

Sections

<table>
<thead>
<tr>
<th>Section</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>PL Concepts</td>
<td>20 pts</td>
</tr>
<tr>
<td>Regular Expressions</td>
<td>8 pts</td>
</tr>
<tr>
<td>Ruby: Fill in the Blanks</td>
<td>7 pts</td>
</tr>
<tr>
<td>Ruby: Coding</td>
<td>16 pts</td>
</tr>
<tr>
<td>OCaml: Typing</td>
<td>4 pts</td>
</tr>
<tr>
<td>OCaml: Debugging</td>
<td>17 pts</td>
</tr>
<tr>
<td>OCaml: Fill in the Blanks</td>
<td>10 pts</td>
</tr>
</tbody>
</table>
General Advice

You can complete answers in any order, and we recommend you look through all of the questions before first so you can gauge how long you should spend on each question. Refer to the counter in the top left corner to ensure you have completed all questions.

Submission

You have 80 minutes to complete this exam (see the timer in the upper right corner for remaining time). Once you begin, you can submit as many times as you want until your time is up. You can even leave this page and come back, and as long as the time hasn't expired, you'll be able to update your submission. This means that if you accidentally submit, refresh, or lose internet temporarily, you'll still be able to work on the test until the time is up. If you come back, click "Resubmit" in the bottom-right corner to resume.

Honor Pledge

Please copy the honor pledge below:

I pledge on my honor that I have not given or received any unauthorized assistance on this examination.

Y

Signature

By entering your name below, you agree that you have read and fully understand all instructions above.

M
**Q2.1** PL Concepts
2 Points

Because Procs allow for functional programming Ruby, Procs are also referentially transparent

- True
- False

*Save Answer*  *Unsaved Changes*

**Q2.2**
2 Points

All dynamically typed languages are latently typed as well

- True
- False

*Save Answer*  *Unsaved Changes*

**Q2.3**
2 Points

Despite Ruby being Dynamically typed, types still have to be deterministic. That is, assuming that `rand(100)` returns a random number from 1 to 100, the following code has an error:

```ruby
def func
  if rand(100) > 50
    a = "hello"
  else
    a = 5
  end
  return a
end
func
```

- True
- False

*Save Answer*  *Unsaved Changes*
Q2.4
2 Points

A Finite State Machine (Finite Automata) can be used to check if an arbitrary string is a palindrome

☐ True
☐ False

Save Answer *Unsaved Changes

Q2.5
2 Points

Ocaml's static type checking means you cannot change variable's type. That is, the following code will throw an error:

```ocaml
let x = 3 in let x = "hello" in x
```

☐ True
☐ False

Save Answer *Unsaved Changes

Q2.6
2 Points

Explain why we only needed concatenation, union, and kleene closure when implementing regex, when there are other symbols like +, ?, {1,2}, etc that exist

Everything else like +, ?, {1,2} are shortcuts for concatenation, union and kleene closure. So if we break a regular expression down to its most basic parts, we only need concat, union, and kleene closure

Save Answer *Unsaved Changes
Q2.7
2 Points

Codeblocks cannot be returned by or passed in to functions

- True
- False

Save Answer *Unsaved Changes

Q2.8
3 Points

Name one advantage a DFA has over an NFA

You will always know which state you are in when traversing a DFA

Save Answer *Unsaved Changes

Q2.9
3 Points

Why would we want to treat functions as data, like we do in OCaml?

Treating functions as data allow for variety of positives:
  - lambda expressions
  - currying
  - code resuability

Save Answer *Unsaved Changes

Q3 Regular Expressions
8 Points

Q3.1
2 Points
Which of the following strings will have a match (partial or exact) with the above regex? Select all that apply.

- 834a
- 11r4
- 863bd7
- 41328

Q3.2 2 Points

Consider the following regex: `/(ga*|bc)*/`

Which of the following strings will have a match (partial or exact) with the above regex? Select all that apply.

- bcga
- ggaa
- bcgaab
- bcag

Q3.3 Regular Expressions 4 Points

Write a regex that will exactly match employee records with the following properties:

- Starts with an employee id which consists of exactly 5 digits followed by a single lowercase or uppercase letter.
- Has a comma and a space after this.
• Has an employee name initials which is either a single uppercase letter followed by one or more lowercase letters or two uppercase letters

employee records:

48931a, Martin
52899B, Alena
43290J, PJ

Examples of invalid emails:

8332f, Mary
48032g, M
46131J, HG
99272g Harrison

\^[a-zA-Z][a-zA-Z]([A-Z][a-z]+[A-Z][2])$
b = /___blank 2___/ 

template = "Player: " + a.source + ", Power: " + b.source 

re = Regexp.new("^"+template+" vs " + template + "\$")

if line =~ re
  if __blank 3__
    puts "#{__4__} wins by #{__5__} power!" #Blank # 4,5
  else
    puts "#{__6__} wins by #{__7__} power!" #Blank # 6,7
  end
end

end

Blank 1

([A-Za-z]*)
([A-Za-z]+)

Blank 2

(\d+)
(\d+)

Blank 3

$2.to_i>$4.to_i
$2.to_i<$4.to_i

Blank 4

$1
$3

Blank 5

$3
$1

Blank 6

$3
$1

Blank 7
Q5 Ruby: Coding
16 Points

It’s kitten season now, and every shelter takes in many kittens everyday. For every cat in the shelter, there will be a record that stores the basic information of this cat (name, age, genders, take-in-date, and available status). To help shelter manage these records, we are going to implement a Ruby program that can read a `record.txt` file and properly store all the read-in information so that they can be reached easily.

For implementing this program, you are given a file named `record.txt` which contains every cat’s information in the shelter, one line for each cat. Each line should be the following format `<Cat Name>, <Take-in date>, <Available Status>.

We define a valid line as follow:
Cat name should start with uppercase letter followed by one or more lowercase letters;
Take-in date is in the format mm/dd/yy, i.e. month, day and year are represented by two digit and separated by `/`;
Available status should be either Available or Adopted

Example of valid input:

```
Anne, 01/01/22, Available
Bob, 02/22/22, Adopted
Cc, 02/02/21, Available
Lucky, 02/03/18, Adopted
```

Example of invalid input:

```
Forest, 01/01/22, Ad
Jonny, 02/02/22, Available
Mika, 2/2/2022, Adopted
```

You will have to implement four functions, described below:

```
initialize(filename): Reads the file and parses the contents. Store the contents in any data structure you like, as long as these other functions work as described below.

get_cat_status_by_name(cat_name): Returns the available status of cat_name. If cat_name does not exist, return nil.
```
take_in_cat_amount_by_month(month): Returns the amount of cats that were taken-in in input month. Return nil if input month is invalid.

max_adopted_cat_month(): Returns the month that most cats are adopted. If no cat is adopted, return nil.

Examples:
Suppose record.txt contains all lines from the example of valid lines.

```ruby
s = CatRecord.new('record.txt')
s.get_cat_status_by_name('Cc')
=> Available
s.take_in_cat_amount_by_month(2)
=> 3
s.max_adopted_cat_month
=> 2
```

class CatRecord
  # Part 1
  def initialize(filename)
    # You may use this block to define your data structures
    # Process each line here!```
  @status={}
  @count=Hash.new(0)

  File.readlines(filename).each do |line|
    re = /([A-Z][a-z]+[\d\d]/d\d\d\d\d\d\d\d\d\d\d, (Available|Adopted)/
    line =~ re
    @status[$1] = $3
    @count[$2.to_i] = @count[$2.to_i]+1
  end
end

  # Part 2
  def get_cat_status_by_name(cat_name)
    @status(cat_name)
  end
```
# Part 3

```ocaml
def take_in_cat_amount_by_month(month)
```

```ocaml
if @count.member?(month) then @count[member]else nil end
```

# Part 4

```ocaml
end
```

```ocaml
# Part 4

def max_adopted_cat_month()
```

```ocaml
max = -1
key = nil
@count.each(|kvl|
  if v > max
    max = v
    key = k
  end
}
key
```

```ocaml
end
```

---

**Q6 Ocaml Typing**

4 Points

- For the following sub-questions, you are **not allowed** to use type annotations
- All pattern matching **must be exhaustive**
- **No other warnings should be raised**

**Q6.1**

2 Points

Write an OCaml expression of type `int option -> int -> bool`
fun x y -> match x with
  Some(v) -> v = y + 1
| None -> false

Q6.2
2 Points

Write an OCaml expression of type \((\text{bool} \rightarrow \text{string}) \rightarrow \text{string option} \rightarrow \text{bool}\)

fun x y -> match y with
  Some(s) -> (x (s = "hello")) ^ "hi" = "hi"
| None -> false

Q7 Ocaml: Debugging
14 Points

Q7.1
4 Points

I want the following expression to return 9. What does it return and how can I fix it?

```ocaml
let x = 4 in
let y = x + 2 in
let x = 3 in
y + 1
```

What does it return?

7

If I wanted to change line 3, what should I change it to?

```ocaml
3 let ____ in
```
y = 8

Q7.2  4 Points
Why does the following expression not compile, and how can I fix it?

```ocaml
let f x = x + 3
let g x = x - 3
let h x = x ^ "3"
[f; g; h];;
```

Why does this not compile?

There is a type error on line 4

If I wanted to change line 3, what could I change it to (with no repeating lines)?

```ocaml
let h x = ____

x *. *3
```

Q7.3  6 Points
Consider the following OCaml code:

```ocaml
type tree = Leaf | Node of int * tree * tree;;
let rec insert t val = match tree with
| Node(x, l, r) ->
| if x = val then Node(x, l, r)
```
There are 5 bugs in the code. Identify 3 and state why it's a bug and the code to fix it.

Bug 1

None exhaustive match, missing match on Leaf
add lines 8 onward
|Leaf -> Node(val,Leaf,Leaf)

Bug 2

Line 6, swapping right and left trees and insert wrong
replace line 6 with
Node(x,insert l val, r) else

Bug 3

Line 7 rewriting entire tree, and again, inserting wrong
Replace 7 with
Node(x,l,insert r val)

Other issues
are that line 7 inserts when it shouldn't
and syntax error on tree and t on line 2

Q8 OCaml: Fill in the Blanks
10 Points

You can use these functions as reference:

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

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

Q8.1
5 Points

Fill in the pattern matching part to finish myfunc which takes in a int * int * string tuple and returns a either the sum or difference of each tuple based on the string.

Examples:

map myfunc [(1,2,"add");(3,4,"sub")] = [3;-1];
map myfunc [] = [];
map myfunc [(1,1,"sub");(-2,1,"add");(5,6,"add")] = [0;-1;11]

You can assume the string is either "add" or "sub"

let myfunc x = match x with
  | (_,"add") -> x + y
  | (_,"sub") -> x - y

Q8.2
5 Points

Consider the following function:

let myfunc2 lst =
foldr (fun (Some x) y -> (string_of_bool x)^y) lst "

If my output is

"truefalsefalsefalsefalse"
Q9 OCaml: Coding
16 Points

Q9.1
8 Points

Given the following functions:

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

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

write a function called `factorial_multiply`, which takes in a list and returns a list of multiplication between the element and the factorial of the element. You may use helper functions and you **may** use the `rec` keyword.

Example:

Ex: `multiply_factorial [1;2;3;4;5] = [1; 4; 18; 96; 600]`

```ocaml
let rec factorial_multiply lst =

let rec factorial x = if x = 0 then 1 else x * factorial (x-1) in
map (fun x -> x * factorial x) lst;;
```

Save Answer  *Unsaved Changes
Write a function called `unflatten` which takes in a `list list` and an `int` and creates a new `list` where each sublist is the size of the `int`. If the length of the list is not a multiple of the `int` then the last sublist can be of smaller size.

Examples

\[
\text{unflatten } [1;2;3;4;5;6] 2 = [[[1;2];[3;4];[5;6]]] \\
\text{unflatten } [1;2;3] 1 = [[1];[2];[3]] \\
\text{unflatten } [1;2;3;4;5] 3 = [[1;2;3];[4;5]]
\]

```ocaml
let unflatten lst x =

let unflatten lst x = let rec mklst l s = if s = 0 then ([] @ l) else match l with
 [] -> ([] @ l) |
 h::l -> let (nlst,rlst) = mklst t (s-1) in ((h::nlst),rlst) in
 let rec uflat l a s = 
 if l = [] then a
 else match (mklst l s) with
 l(n,r) -> uflat r (a @ [n]) s in
 uflat lst [] x::;
```

**Q10 FSMs**

4 Points

Consider the following FSM

![Finite State Machine Diagram](image-url)
Q10.1
1 Point

Is the String "abcaab" accepted?

☐ No
☐ Yes

Save Answer *Unsaved Changes

Q10.2
3 Points

Describe the set of strings this machine accepts (Regex or words)

a+((c+)(b+)a+)/((b+)a+)

Save Answer *Unsaved Changes

Save All Answers Submit & View Submission