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

Section	Points
PL Concepts	**[20 pts]**
Regular Expressions	**[8 pts]**
Ruby: Fill in the Blanks	**[7 pts]**
Ruby: Coding	**[16 pts]**
OCaml: Typing	**[4 pts]**
OCaml: Debugging	**[17 pts]**
OCaml: Fill in the Blanks	**[10 pts]**

Section	Points
OCaml: Coding	[**16 pts**]
FSM	[**4 pts**]

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.

Ye

Signature

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

Me

Save Answer ***Unsaved Changes**

Q2 PL Concepts

20 Points

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:

```
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 epxression 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 Expresions

8 Points

Q3.1

2 Points

Consider the following regex: `/^[0-9]?[0-5]{2,3}[a-z]*/`

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

- ☑ 834a
- ☑ 11r4
- □ 863bd7
- ☑ 41328

Save Answer *Unsaved Changes

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

Save Answer *Unsaved Changes

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, Allena
43290j, PJ
```

Examples of invalid emails:

```
8332f, Mary

48032g, M
46131, HG
99272g Harrison
```

```
^\d{5}[a-zA-Z], ([A-Z][a-z]+|[A-Z]{2})$
```

Save Answer ***Unsaved Changes**

Q4 Ruby: Fill in The Blank

7 Points

Fill in the blank to complete the implementation `get_winner` which takes in a string input and should print the winner of 2 players given their power which is an integer.

Each player in the input string will be a single word with any amount of lower and uppercase characters in any order. Each player will then be followed by a number of any length that represents that player's power rating.

Examples

```
get_winner("Player: Saber, Power: 5000 vs PLayer: Caster, Power: 2500")
#prints "Saber wins by 2500 power!"
get_winner("Player: Salter, Power: 9999 vs Player: Rider, Power: 10000")
#prints "Rider wins by 1 power!"
```

Implmentation:

```
def get_winner(line)
  a = /___blank 1___/
```

```
    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

$1
$3

Save Answer ***Unsaved Changes**

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.

```
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, (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
  def take_in_cat_amount_by_month(month)
```

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

```
  end

  # Part 4
  def max_adopted_cat_month()
```

```
max = -1
key = nil
@count.each{|k,v|)
 if v > max
  max = v
  key = k
 end
}
key
```

```
  end
end
```

Save Answer *Unsaved Changes

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
```

Save Answer ***Unsaved Changes**

Q6.2

2 Points

Write an OCaml expression of type `(bool -> string) -> string option -> bool`

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

Save Answer ***Unsaved Changes**

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?

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

What does it return?

7

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

```
3 let ____ in
```

y = 8

Save Answer *Unsaved Changes

Q7.2

4 Points

Why does the following expression not compile, and how can I fix it?

```
1 let f x = x + 3 in
2 let g x = x - 3 in
3 let h x = x ^ "3" in
4 [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)?

```
3 let h x = ____
```

x * 3

Save Answer *Unsaved Changes

Q7.3

6 Points

Consider the following OCaml code:

```
1 type tree = Leaf|Node of int * tree * tree;;

2 let rec insert t val = match tree with
3 |Node(x,l,r) ->
4  if x = val then Node(x,l,r)
```

```
5  else if x < val then
6    insert Node(x,r,l) val else
7    Node(val,Leaf,Leaf);;
```

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 exhuastive match, missing match on Leaf
add lines 8 onward
|Leaf -> Node(val,Leaf,Leaf)

Bug 2

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

Bug 3

Line 7 rewritting 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

Save Answer ***Unsaved Changes**

Q8 OCaml: Fill in the Blanks

10 Points

You can use these functions as reference:

```
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
|(* blank, but you can list many patterns *)
|_ -> failwith "error"
in map myfunc lst;;
```

(x,y,"add") -> x + y
|(x,y,"sub") -> x - y

Save Answer ***Unsaved Changes**

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

```
"truefalsetruefalsefalse"
```

what is my input?

```
[Some(true);Some(false);Some(true);Some(false);Some(false)]
```

Save Answer ***Unsaved Changes**

Q9 OCaml: Coding

16 Points

Q9.1

8 Points

Given the following functions:

```
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]`

```
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**

Q9.2

8 Points

Write a function called `unflatten` which takes in a `'a list` and an `int` and creates a `'a list 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

```
unflatten [1;2;3;4;5;6] 2 = [[1;2];[3;4];[5;6]]
unflatten [1;2;3] 1 = [[1];[2];[3]]
unflatten [1;2;3;4;5] 3 = [[1;2;3];[4;5]]
```

```
let unflatten lst x =
```

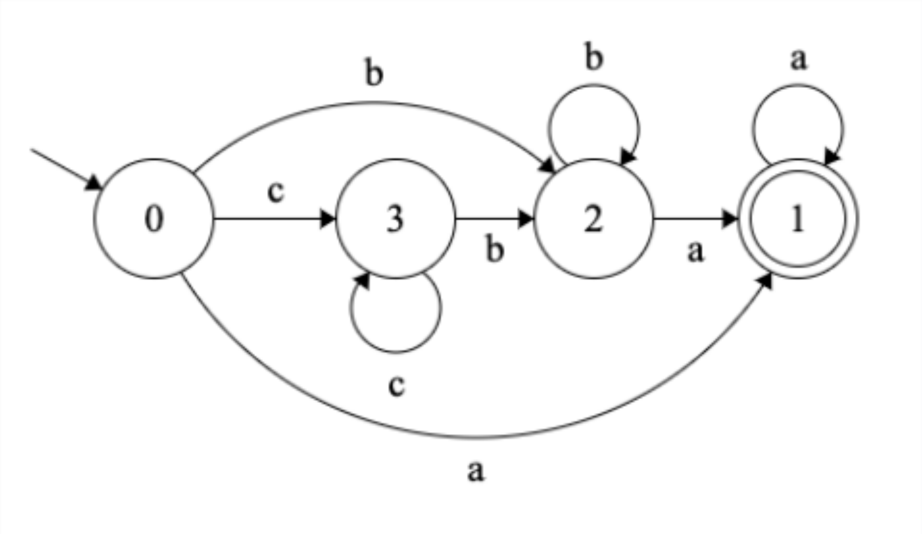
```
let unflatten lst x = let rec mklst l s = if s = 0 then ([],l) else match l with
[]->([],l)
|h::t -> let (nlst,rlst) = mklst t (s-1) in ((h::nlst),rlst) in
let rec uflat l a s =
if l = [] then a
else match (mklst l s) with
|(n,r) -> uflat r (a@[n]) s in
uflat lst [] x;;
```

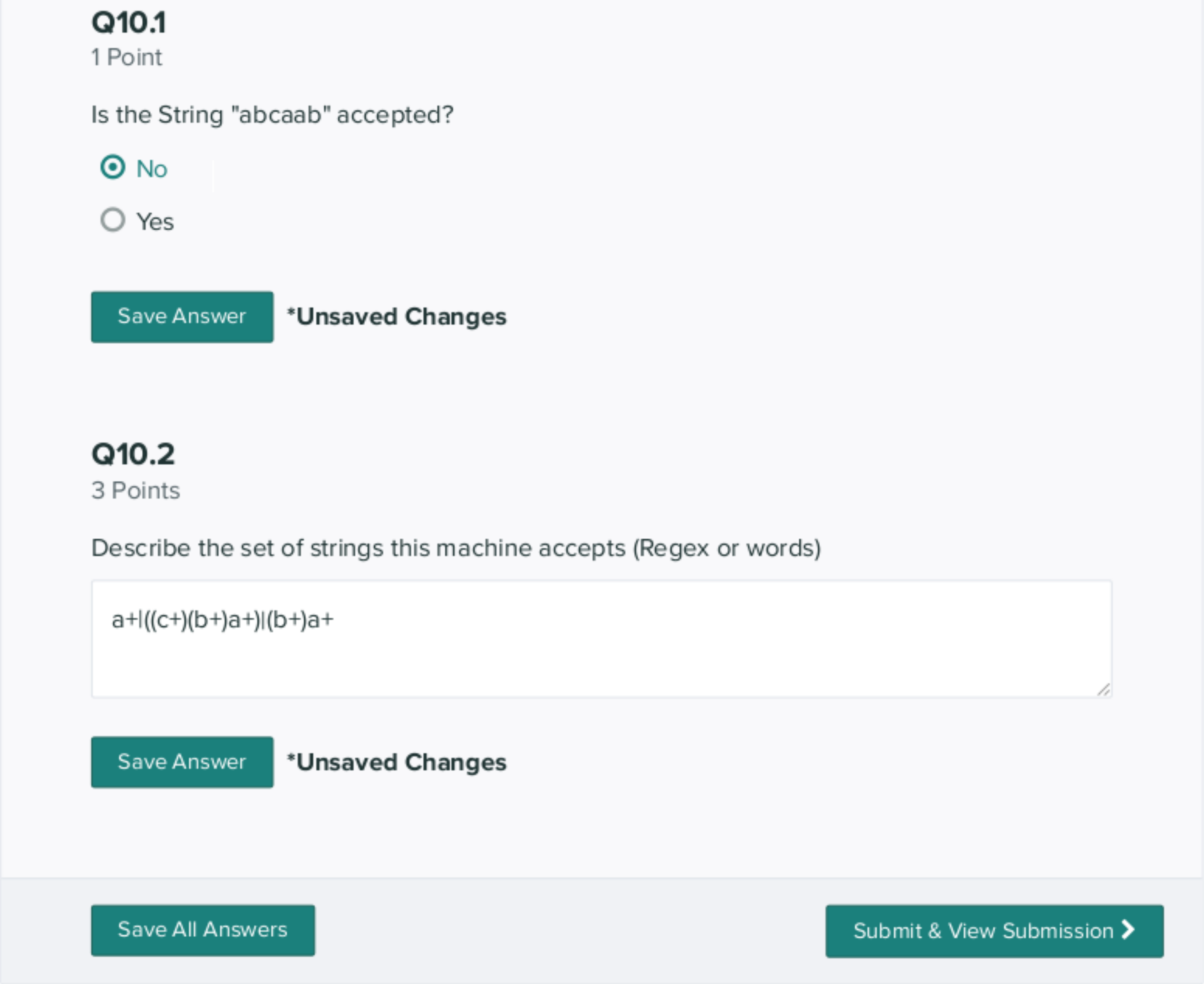
Save Answer ***Unsaved Changes**

Q10 FSMs

4 Points

Consider the following FSM





Q10.1

1 Point

Is the String "abcaab" accepted?

● No

○ Yes

Q10.2

3 Points

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

a+|((c+)(b+)a+)|(b+)a+