Exam 1 from Spring 2021 (Practice)

STUDENT NAME

Search students by name or email...

Q1 Introduction

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`. 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

- PL Concepts [11pts]
- Regular Expressions [9pts]
- Ruby: What's the Input? [14pts]
- Ruby Code: Fill-in-the-Blank [14pts]
- Ruby Coding [6pts]
- OCaml Typing [10pts]
- OCaml: What's the Input? [12pts]
- OCaml Fill-In-The-Blank [14pts]
- OCaml Coding [10pts]

General Advice

You can complete answers in any order, and we urge you to look through all of the questions at the beginning so you can accurately 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.

Enter your answer here

Signature

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

Enter your answer here

Save Answer

Q2 PL Concepts

11 Points

The following true/false and multiple-choice questions test your knowledge of a variety of programming language concepts.

Q2.1 Lists in Ruby and OCaml

1 Point

Ruby arrays and OCaml lists both may only contain elements that are all the same type.

- ○ True
- ○ False

Save Answer

Q2.2 OCaml Behavior

2 Points

What's the best way of describing what's happening on line 2 of the following code?

```
(*1*)  let rec f x =
(*2*)      let x = x-1 in
(*3*)      if x = 0 then 0 else f x
```

- ○ It is updating f's parameter x's value
- ○ It is shadowing f's parameter x
- ○ It is testing whether x is equal to x-1
- ○ It is making a recursive call on x

Save Answer

Q2.3 Dynamically Typed Languages

1 Point

In dynamically typed languages, there may be type errors that are not caught until the code is executed.

- ○ True
- ○ False

Save Answer

Q2.4 Mutability

4 Points

Which of the following are mutable? Select all that apply.

- ☐ OCaml variables
- ☐ OCaml lists
- ☐ Ruby variables
- ☐ Ruby arrays

Save Answer

Q2.5 Dynamic vs. Static Typing

1 Point

Suppose the language DynCaml has the *exact same behavior* (and syntax) as normal OCaml, *except* that it is dynamically typed (rather than statically typed). Which of the following code snippets would be allowed, and run successfully, in DynCaml but not OCaml?

- ○ `let x = 1::[1; 2; 3] in 0`
- ○ `let x = 1 in if x > 0 then "greater" else -1`
- ○ `let rec f x y = x + y in f 1 "hello"`
- ○ `let rec f x = if x>0 then f (x-1) else 0`

Save Answer

Q2.6 Closures

1 Point

What variables must be in the environment of the closure for function `f` in the following code?

```
let foo x y =
  let f z = x + y + z in
  let a = x + y in
  a + (f b)
```

- x and y
- x, y, and z
- x, y, z, and b
- x, y, a, z, and b

Save Answer

Q2.7 Property-based Testing
1 Point

How is property-based testing (PBT) different from normal unit testing?

- A single property is used to test many automatically generated inputs, not a single hand-crafted one
- A property is sure to hold for all possible inputs, whereas a unit test holds for just one
- In unit testing, generators and shrinkers are written by hand
- You can always write properties that completely capture what code is supposed to do

Save Answer

Q3 Regular Expressions
9 Points

The following problems ask to write or talk about regular expressions for matching input patterns.

Q3.1 Regex translation
3 Points

Rewrite the following regular expression

```
[abcd]+
```

so that does it not use character classes or the plus (`+`) operator, but matches exactly the same strings.

Enter your answer here

Save Answer

Q3.2 2-D Coordinates
6 Points

Write a regular expression that matches a 2-D decimal coordinate. The decimal numbers should range from 0.0 - 99.99. The decimal point will always be present, and there can be one or two

digits on either side of it. There should be no whitespace.

These strings should match

```
(1.0,2.66)
(0.00,00.0)
(10.01,0.10)
```

but not strings like this:

```
(1.0, 2.0)
(1.0,2.666)
(1.11,2)
(.2,0.0)
(1.,2.0)
```

Enter your answer here

Save Answer

Q4 Ruby: What's the Input?

14 Points

For these questions, you are shown some Ruby code along with an execution of it that produces a particular output. Your job is to figure out what *input* could produce that output. (There are no syntax errors in the code given.)

Q4.1 arrays, eq

4 Points

Consider the following code

```
puts xxxx[0]
puts xxxx[2]
puts xxxx == yyyy
```

To what can we set `xxxx` and `yyyy` at the start so that the following is printed?

```
1
13
true
```

xxxx

Enter your answer here

yyyy

Enter your answer here

Save Answer

Q4.2 codeblocks

4 Points

Consider the following Ruby code

```
def m(z)
  if z == 0
    puts yield z
  else
    puts yield (z+1)
  end
end
m(xxxx) { |a| a+12 }
m(yyyy) { |b| b*3 }
```

To what can we set `xxxx` and `yyyy` at the start so that the following is printed?

```
12
12
```

xxxx

Enter your answer here

yyyy

Enter your answer here

Save Answer

Q4.3 classes

6 Points

Consider the following Ruby code

```
class Note
  @@all = []
  def initialize(note)
    @@all.push(note)
    @note = note
  end
  def update(v)
    @@all.push(v)
    @note = v
  end
  def to_s
    @@all.length.to_s + "," + @note
  end
end

n1 = Note.new("dog")
if xxxx then
  n1.update(yyyy)
```

```
end
puts n1
n2 = Note.new(zzzz)
puts n2
```

To what can we set `xxxx`, `yyyy`, and `zzzz` at the start so that the following is printed?

```
2,wolf
3,cat
```

xxxx

Enter your answer here

yyyy

Enter your answer here

zzzz

Enter your answer here

Save Answer

Q5 Ruby Code: Fill-in-the-Blank

14 Points

The next three questions contain partial programs written in Ruby. Complete each program so that it produces the given output.

Q5.1 grades

2 Points

Consider the following Ruby code

```
grades = { "Bob" => 4, "Chris" => 3 }
grades ____________________ = 2
sum = 0
grades.keys.each {|k| sum = sum + k.length }
puts sum
```

To what can we fill in the blank so that the following is printed?

```
13
```

Enter your answer here

Save Answer

Q5.2 total_grades

8 Points

`total_grades` takes a string input and should print the total number of points for two assignments for the student specified by that input. Example calls:

```
total_grades("ID: alice, Scores: 92 08")
# prints "alice got 100 points"

total_grades("ID: brodriguez12, Scores: 32 25")
# prints "brodriguez12 got 57 points"
```

Each ID in the input string will start with a lowercase letter, and then is followed by any number of lowercase letters or numeric digits. The spacing shown in the above examples is precise: No more and no less should be permitted. The keywords `ID` and `Scores` should be matched exactly. There will always be two (non-negative) scores, each of which is exactly two digits.

Fill in the four labeled blanks to complete this implementation.

```
def total_grades(line)
  if line =~ /^ID: ________, Scores: ________$/ # 1, 2
    puts "#{________} got #{________} points" # 3, 4
  end
end
```

#1

Enter your answer here

#2

Enter your answer here

#3

Enter your answer here

#4

Enter your answer here

Save Answer

Q5.3 hash_to_1d_array

4 Points

`hash_to_1d_array` takes a hash `map` and converts it into a one-dimensional array, with elements in sorted order. Example calls:

```
hash_to_1d_array({})
# returns []
```

```
hash_to_1d_array({0=>2, 3=>12, 32=>1})
# returns [0, 1, 2, 3, 12, 32]

hash_to_1d_array({8=>7, 6=>5, 1=>2})
# returns [1, 2, 5, 6, 7, 8]
```

Fill in the two labeled blanks to complete this implementation.

```
def hash_to_1d_array(map)
  b = []
  map.each {____________________} # 1
  ____________________ # 2
  return b
end
```

#1

Enter your answer here

#2

Enter your answer here

Save Answer

Q6 Ruby Coding

6 Points

Implement method `combine` which takes two hashes `h1` and `h2` and incorporates all of `h2`'s key-value bindings into `h1` (modifying `h1`), as follows:

- If `h2[k]` maps to `v2` and `h1[k]` does *not* exist, then add a mapping from `k` to `v2` in `h1`
- If `h2[k]` maps to `v2` and `h1[k]` *does* exist and maps to `v1`, then update the mapping in `h1` for `k` to be to `v1 + v2`.

For example:

```
h1 = { "a" => 10, "b" => 20 }
h2 = { "b" => 30, "c" => 40}
combine(h1,h2)
puts h1.inspect
# prints {"a"=>10, "b"=>50, "c"=>40}
```

Write your code here:

Enter your answer here

Save Answer

Q7 OCaml Typing

10 Points

Each of the following questions asks you to write an OCaml expression that has the given type. **Do not use type annotations**.

Q7.1 (int * int) list

2 Points

Without using type annotations, write an OCaml expression that has type `(int * int) list`

Enter your answer here

Save Answer

Q7.2 int list -> bool

3 Points

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

Enter your answer here

Save Answer

Q7.3 (int -> 'a) -> 'b -> 'a * 'b

3 Points

Without using type annotations, write an OCaml expression that has the type `(int -> 'a) -> 'b -> 'a * 'b`.

Enter your answer here

Save Answer

Q7.4 int -> int

2 Points

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

```
((fun x -> (fun y -> x + y)) _____________ )
```

Enter your answer here

Save Answer

Q8 OCaml: What's the Input?

12 Points

For these questions, you are shown some OCaml code along with an execution of it that produces a particular output. Your job is to figure out what input could produce that output. (There are no syntax or type errors in the code given.)

Q8.1 recursion

3 Points

```
let rec foo n =
  if n = 0 then true
  else
    if n = 1 then false
    else foo (n-2)
```

Suppose that

```
foo (1 + zzzz) = true
```

What is a *non-negative* `int` you can use for `zzzz`?

Enter your answer here

Save Answer

Q8.2 map

3 Points

Recall that `List` module's `map` function is defined as follows.

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

Suppose that

```
map (fun x -> (x,x)) zzzz = [(1,1);(2,2)]
```

What is `zzzz`?

Enter your answer here

Save Answer

Q8.3 partial application

3 Points

Suppose we have

```
let f a b = a - b in
let g = f 10 in
let a = 4 in
g zzzz
```

What must `zzzz` be so that this expression evaluates to `2` (i.e., `g zzzz = 2`)?

Enter your answer here

Save Answer

Q8.4 first-class functions

3 Points

Consider the following function `g`:

```
let rec g f x =
  if (f x) = 0 then
    x
  else
    0
```

Suppose that `g zzzz 1 = 1`.

What is `zzzz`? (There are many right answers.)

Enter your answer here

Save Answer

Q9 OCaml Fill-In-The-Blank

14 Points

The problems here will show you partial implementations of OCaml functions. Complete each implementation by filling in the blanks.

Q9.1 parity with fold

5 Points

The function `parity x l` should return `true` if `x` occurs inside list `l` *an even number of times* (which includes 0). For example

```
parity 5 [5;5;1;0] = true
parity 3 [5;1;0]   = true
parity 4 [0;5;4]   = false
```

Complete the implementation of `parity` by writing what goes in the blank.

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

Answer

Enter your answer here

Save Answer

Q9.2 tail recursion

5 Points

The following function is *not* tail recursive:

```
let rec f y =
  if y = 0 then 1
  else y * f (y/2)
```

Complete the implementation of `f'` below, which is a tail recursive version of `f`, by filling in the two blanks:

```
let f' y =
  let rec aux x a =
    if x = 0 then a
    else _________________ in (* 1 *)
  aux _____________ (* 2 *)
```

1

Enter your answer here

2

Enter your answer here

Save Answer

Q9.3 calc

4 Points

The following code defines a data type `t` as either an `int` or `string`, where the `add` function "sums" two `t` elements---if they are two `int`s it uses integer addition, otherwise it uses string concatenation. Examples:

```
add (Int 1)      (Int 2)              = Int 3
add (String "1") (String " is not 2") = String "1 is not 2"
add (Int 1)      (String " is not 2") = String "1 is not 2"
add (String "1 is not ")      (Int 2) = String "1 is not 2"
```

Complete the function by filling in the four blanks.

```
type t =
  Int of int
| String of string

let add x y =
  match (x,y) with
    (Int i, Int j)      -> Int (i+j)
  | (_______, String j) -> _________ (* 1, 2 *)
  | (Int i, String j)   -> String ((string_of_int i)^j)
  | ____________________ -> _________ (* 3, 4 *)
```

1

Enter your answer here

2

Enter your answer here

3

Enter your answer here

4

Enter your answer here

Save Answer

Q10 OCaml Coding

10 Points

Complete solutions to the following problems in OCaml. You are welcome to use `fold_left` (the same as the `fold` function shown earlier), `fold_right`, `map`, `mem`, or other functions from the `List` module, or you are welcome to write your code entirely, including (recursive) helper functions.

Q10.1 proc

4 Points

Write a function `proc` that takes an `bool` and then returns a function. The returned function takes a pair of `int`s and returns their sum if the original `bool` was `true` and returns their difference otherwise. For example:

```
let f = proc true;;
f (1,2) = 3
f (3,7) = 10
```

whereas

```
let g = proc false;;
g (1,2) = -1
g (3,7) = -4
```

Solution here:

Enter your answer here

Save Answer

Q10.2 sum_exists

6 Points

Write a function called `sum_exists` which takes a list of integers and a target integer, and returns `true` if the list contains two elements, whose sum is equal the target integer. Return false otherwise. (So, the type of `sum_exists` is `int list -> int -> bool`.)

For example:

```
sum_exists [1; 2; 3; 4; 5] 8 = true
sum_exists [1; 2; 3; 4] 5    = true
sum_exists [8; 10] 11        = false
sum_exists [8; 2; 8; 1] 10   = true
```

Solution here:

Enter your answer here

Save Answer

Save All Answers

Submit & View Submission ›