QUESTION 2 PL Concepts
The following true/false and multiple-choice questions test your knowledge of a variety of programming language concepts.

Q2.1
Ruby arrays and OCaml lists both may only contain elements that are all the same type.

False

Q2.2
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 shadowing f's parameter x

Q2.3
In dynamically typed languages, there may be type errors that are not caught until the code is executed.

True

Q2.4
Which of the following are mutable? Select all that apply.

Ruby variables, Ruby arrays

Q2.5
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 in if x > 0 then "greater" else -1
```
Q2.6
What variables must be in the environment of the closure for function f in the following code?

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

Q2.7
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

QUESTION 3 Regex
The following problems ask to write or talk about regular expressions for matching input patterns.

Q3.1
Rewrite the following regular expression so that it does not use character classes or the + operator, but matches exactly the same strings.

```
[abcd]+
(a|b|c|d)(a|b|c|d)*
```
Q3.2
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, 2.0)
(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)

\(\d{1,2}\. \d{1,2}, \d{1,2}\. \d{1,2}\)
QUESTION 4 What’s the Input?

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
Consider the following code

```ruby
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
```

```ruby
xxxx: [1, nil, 13] or {0=>1, 2=>13}
yyyy: [1, nil, 13] or xxxx
```

Q4.2
Consider the following Ruby code

```ruby
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
```

```ruby
xxxx: 0
yyyy: 3
```
Q4.3
Consider the following Ruby code

```ruby
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: true or any truthy value
yyyy: "wolf"
zzzz: "cat"
QUESTION 5 Ruby Fill in the Blank

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

Q5.1
Consider the following Ruby code

```ruby
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
```

```
[ "abcde"] (or any string of length 5 in square brackets)
```
Q5.2

`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:

```ruby
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.

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

#1: `[a-z][a-z0-9]*`
#2: `\d{2} \d{2}`
#3: `$1`
#4: `$2.to_i + $3.to_i`

(5 Points) - Reasonable, 3 for the regex, 2 for the second part
Q5.3

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

```ruby
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.

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

#1: |k,v| b.append([k,v])
#2: b.sort!
QUESTION 6 Ruby Coding

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:

```ruby
def combine(h1, h2)
  h2.each do |k, v|
    if h1[k]
      h1[k] += v
    else
      h1[k] = v
    end
  end
end
```

```ruby
h1 = { "a" => 10, "b" => 20 }
h2 = { "b" => 30, "c" => 40 }
combine(h1, h2)
puts h1.inspect  # prints{"a"=>10, "b"=>50, "c"=>40}
```
QUESTION 7 OCaml Typing

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

**Q7.1**
Without using type annotations, write an OCaml expression that has the type 
(int * int) list

$$[1,2]$$
$$[(1,2); (3,4)]$$

**Q7.2**
Without using type annotations, write an OCaml expression that has the type
int list -> bool

```ocaml
fun lst -> lst = [2]
```

**Q7.3**
Without using type annotations, write an OCaml expression that has the type
(int -> 'a) -> 'b -> 'a * 'b

```ocaml
fun f x -> (f 3, x)
```

**Q7.4**
Without using type annotations, write an OCaml expression to fill in the blank so that entire expression has type int -> int

```ocaml
((fun x -> (fun y -> x + y)) ____________ )
```

0 or any integer
QUESTION 8 OCaml: What’s the Input?

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**

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

Suppose that

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

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

1 (or any positive odd integer)

**Q8.2**

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

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

Suppose that

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

What is zzzz?

[1;2]
Q8.3
Suppose we have

```ocaml
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)?

8

Q8.4
Consider the following function g:

```ocaml
let rec g f x =
  if (f x) = 0 then
    x
  else
    0
```
Suppose that g zzzz 1 = 1. What is zzzz?

zzzz: `(fun x -> x - 1)`
QUESTION 9 OCaml Fill-in-the-Blank

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

Q9.1
The function `parity x l` should return `true` if `x` occurs inside list `l` an even number of times (which includes 0). For example

<table>
<thead>
<tr>
<th><code>parity</code></th>
<th><code>l</code></th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>[5;5;1;0]</td>
<td>true</td>
</tr>
<tr>
<td>3</td>
<td>[5;1;0]</td>
<td>true</td>
</tr>
<tr>
<td>4</td>
<td>[0;5;4]</td>
<td>false</td>
</tr>
</tbody>
</table>

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

```ocaml
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

(fun a e -> if e = x then not a else a)
```

Q9.2
The following function is *not* tail recursive:

```ocaml
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:

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

1: aux (x/2) (a*x)
2: y 1```
Q9.3

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 ints 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: String i
2: String (i ^ j)
3: (String i, Int j)
4: String (i ^ (string_of_int j))
**QUESTION 10 OCaml Coding**

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

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

whereas

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

```ocaml
let proc b (x,y) = if b then x+y else x-y
```

**Q10.2**
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 to the target integer. Return false otherwise. (So, the type of sum_exists is int list -> int -> bool.)

For example:

```ocaml
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
```

```ocaml
let rec sum_exists lst x =
    match lst with
    | [] -> false
    | h::t -> (List.mem (x-h) t) || (sum_exists t x)
```