LAMBDAS

COMPUTER SCIENCE MENTORS CS 88

February 22 to 26

1 Lambdas

A lambda expression evaluates to a function, called a lambda function. For example, `lambda x, y: x + y` is a lambda expression, and can be read as "a function that takes in two parameters `x` and `y` returns `x + y`."

may take in multiple params, but only can return one expr.

A lambda expression by itself evaluates to a function but does not bind it to a name. Also note that the return expression of this function is not evaluated until the lambda is called. This is similar to how defining a new function using a def statement does not execute the function's body until it is later called.

```
>>> what = lambda x : x + 5
>>> what
<function <lambda> at 0xf3f490>
```

Unlike `def` statements, lambda expressions can be used as an operator or an operand to a call expression. This is because they are simply one-line expressions that evaluate to functions.

```
>>> (lambda y: y + 5)(4)
9
>>> (lambda f, x: f(x))(lambda y: y + 1, 10)
11
```

lambda expr.	`(lambda y: y + 5)(4)`	`(lambda f, x: f(x))(lambda y: y + 1, 10)`

def statement	def f(y):
	  return y + 5
	f(4)
		def g(f, x):
		  return f(x)
		def h(y):
		  return y + 1
		g(h, 10)

note

```
(lambda f, x: f(x))(lambda y: y + 1, 10)
```

↓ equivalent to

```
def g(f, x):
    return f(x)

g(lambda y: y+1, 10)
```

2 separate elements: the lambda function, and the number 10. these 2 elements are passed in as arguments to the previous lambda function with f being bound to lambda y: y + 1 and x being bound to the number 10.

common confusion

(lambda y: y+1, 10) does not mean that there is 1 lambda function that takes in y and then returns both y+1 and 10. to return more than 1 thing from a lambda function, you would use parentheses to indicate what you're returning, for example ⟶ lambda y: (y+1, 10)

1. What do lambda expressions do? Can we write all functions as lambda expressions? (Hint: think about the limitations of lambdas) In what cases are lambda expressions useful?

function w/o a name, must save to variable
useful in map function

2. Determine if each of the following will error:

```
>>> 1/0
```
Error

<func <lambda> at 0x.... >

```
>>> boom = lambda: 1/0
```
no error

```
>>> boom()
```
Error

3. Express the following lambda expression using a **def** statement, and the **def** statement using a lambda expression.

```
pow = lambda x, y: x**y
```

```
def pow(x, y):
    return x ** y
```

```
def foo(x):
    def f(y):
        def g(z):
            return x + y * z
        return g
    return f
```

g = lambda z: x + y * z

f = lambda y: g

foo = lambda x: f

foo = lambda x: lambda y: lambda z: x+y*z

g

f

4. For each of the following lines of code, determine what would be printed as the output.

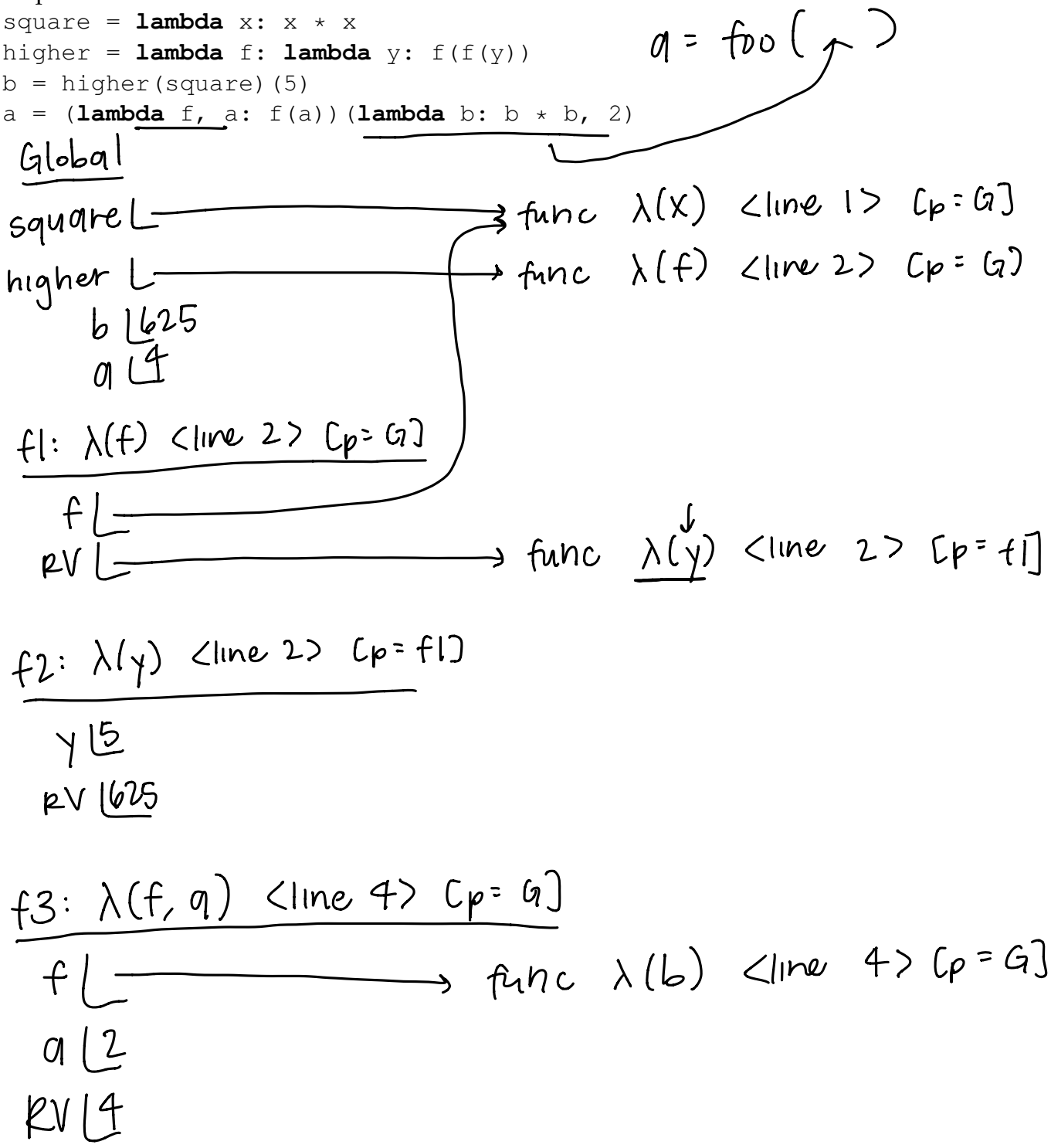
```
>>> plus_one = lambda i: print(i+1)
>>> plus_one
<func <lambda> at 0x...>
>>> plus_one(6)
7
>>> multiply = lambda x, y: x*y
>>> harder_lambda = lambda func: print(func(4, 5))
>>> harder_lambda(multiply)
multiply
4*5 = 20
20
```

5. What would Python print?

```
>>> a = lambda: 5
>>> a()

>>> a(5)

>>> b = lambda: lambda x: 3
>>> b()(15)

>>> c = lambda x, y: x + y
>>> c(4, 5)

>>> d = lambda x: lambda y: x * y
>>> d(3)

>>> d(3)(3)

>>> e = d(2)
>>> e(5)

>>> f = lambda: print(1)

>>> g = f()
```

6. **Challenge Problem**: Draw Environment Diagrams for the following lines of code. Note: When working with lambdas in environment diagram problems, it is really helpful to write down which line the lambda was defined on.

```
1  square = lambda x: x * x
2  higher = lambda f: lambda y: f(f(y))
3  b = higher(square)(5)
4  a = (lambda f, a: f(a))(lambda b: b * b, 2)
```

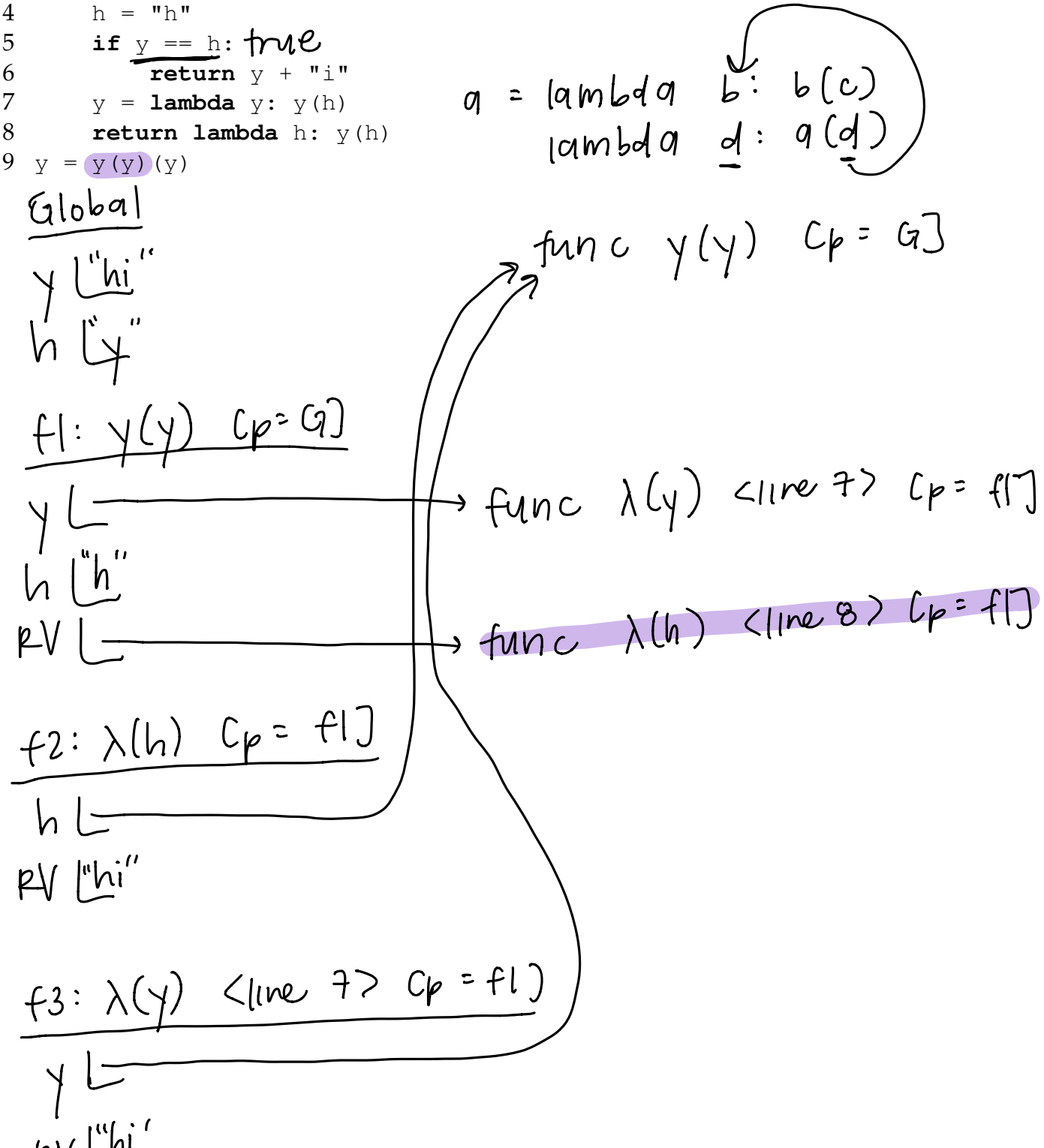


7. *The following question is extremely difficult. Something like this would not appear on the exam. Nonetheless, it's a fun problem to try.*

   Draw the environment diagram that results from executing the code below.

   Note that using the + operator with two strings results in the second string being appended to the first. For example `"C" + "S"` concatenates the two strings into one string `"CS"`

```
1 y = "y"
2 h = y
3 def y(y):
4     h = "h"
5     if y == h:
6         return y + "i"
7     y = lambda y: y(h)
8     return lambda h: y(h)
9 y = y(y)(y)
```



RV ["hi"

f4: y(y) [p = 9]

y ["h"

h ["h"

RV ["hi"

1. What do lambda expressions do? Can we write all functions as lambda expressions? (Hint: think about the limitations of lambdas) In what cases are lambda expressions useful?

   a: create functions

   b: no, complex multi-line functions can't be written using lambdas

2. Determine if each of the following will error:

```
>>> 1/0
```
Error
```
>>> boom = lambda: 1/0
```
OK - only defining function, not going into body
```
>>> boom()
```
Error

3. Express the following lambda expression using a **def** statement, and the **def** statement using a lambda expression.

```
pow = lambda x, y: x**y
```

```
def pow(x,y):
    return x ** y
```

```
def foo(x):
    def f(y):
        def g(z):
            return x + y * z
        return g
    return f
```

```
foo = lambda x: lambda y: lambda z: x+y*z
```
g

f

lambda functions return whatever comes right after the colon:

foo returns f, f returns g

4. For each of the following lines of code, determine what would be printed as the output.

```
>>> plus_one = lambda i: print(i+1)
>>> plus_one
<function <lambda> at ... >
>>> plus_one(6)
7
>>> multiply = lambda x, y: x*y
>>> harder_lambda = lambda func: print(func(4, 5))
>>> harder_lambda(multiply)
20
```

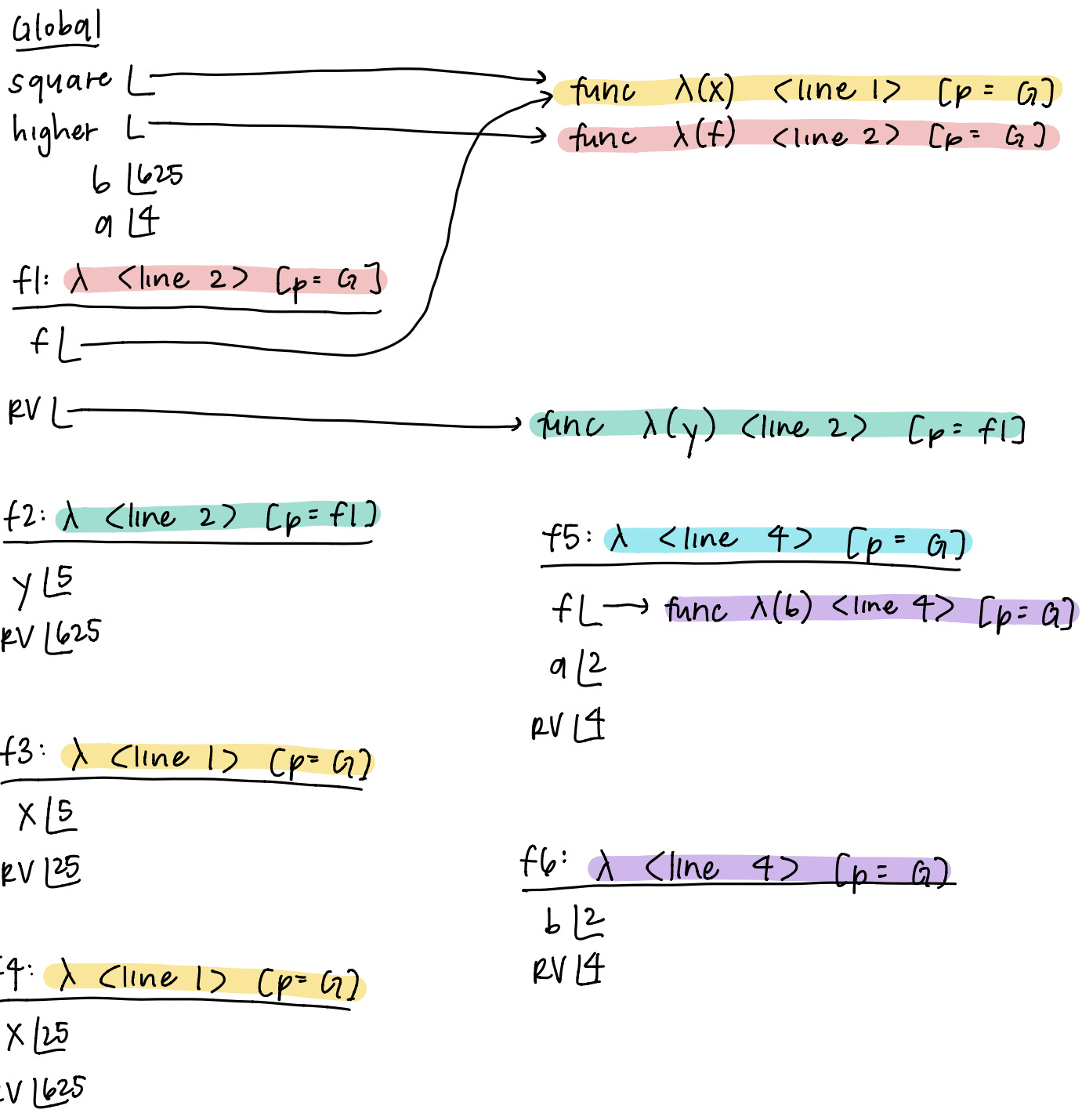
*Repeat question from discussion

5. What would Python print?

```
>>> a = lambda: 5
>>> a()
```
5
```
>>> a(5)
```
Error - wrong number of arguments
```
>>> b = lambda: lambda x: 3
>>> b()(15)
```
3
```
>>> c = lambda x, y: x + y
>>> c(4, 5)
```
9
```
>>> d = lambda x: lambda y: x * y
>>> d(3)
```
function <lambda> at ...
```
>>> d(3)(3)
```
9
```
>>> e = d(2)
>>> e(5)
```
10
```
>>> f = lambda: print(1)
```
← no output
```
>>> g = f()
```
1

6. **Challenge Problem**: Draw Environment Diagrams for the following lines of code. Note: When working with lambdas in environment diagram problems, it is really helpful to write down which line the lambda was defined on.

```
1 square = lambda x: x * x
2 higher = lambda f: lambda y: f(f(y))
3 b = higher(square)(5)
4 a = (lambda f, a: f(a))(lambda b: b * b, 2)
```



7. *The following question is extremely difficult. Something like this would not appear on the exam. Nonetheless, it's a fun problem to try.*

   Draw the environment diagram that results from executing the code below.

   Note that using the + operator with two strings results in the second string being appended to the first. For example `"C" + "S"` concatenates the two strings into one string `"CS"`

```
1 y = "y"
2 h = y
3 def y(y):
4     h = "h"
5     if y == h:
6         return y + "i"
7     y = lambda y: y(h)
8     return lambda h: y(h)
9 y = y(y)(y)
```