1 Higher-Order Functions

1.1 Draw the environment diagram that results from running the code.

```python
x = 20
def foo(y):
    x = 5
def bar():
    return lambda y: x - y
return bar

y = foo(7)
z = y()
print(z(2))
```

https://goo.gl/i2yiQF

1.2 What’s the difference here?

```python
x = 20
def bar():
    return lambda y: x - y
def foo(y):
    x = 5
    return bar

y = foo(7)
z = y()
print(z(2))
```

The parent of `bar` is now the global frame so we lookup the value of `x` from there.

1.3 Why and where do we use lambda and higher-order functions?

In practice, we use lambda functions and higher-order functions to write short `adapters` programs, or functions that help us connect two programs together. In the `Maps` project, you’ll have an opportunity to `adapt` a particular problem, predicting user ratings, to general machine learning algorithms.
Consider the following method.

```python
def make_adder(x):
    def adder(n):
        return x + n
    return adder

make_adder(4)(5)
```

(a) What is the operator of the above expression?

```python
make_adder(4)
```

(b) What are the operands?

```python
5
```

(c) Draw the expression tree.

```
make_adder(4)(5)
       /     \
make_adder(4)  5
       /     \
func make_adder(x)  4
```
1.5 Write a higher-order function that passes the following doctests.

*Challenge:* Write the function body in one line.

```python
def mystery(f, x):
    """
    >>> from operator import add, mul
    >>> a = mystery(add, 3)
    >>> a(4) # add(3, 4)
    7
    >>> a(12)
    15
    >>> b = mystery(mul, 5)
    >>> b(7) # mul(5, 7)
    35
    >>> b(1)
    5
    >>> c = mystery(lambda x, y: x * x + y, 4)
    >>> c(5)
    21
    >>> c(7)
    23
    """

    def helper(y):
        return f(x, y)
    return helper

Challenge solution:

```python
    return lambda y : f(x, y)

1.6 What would Python display?

```python
>>> foo = mystery(lambda a, b: a(b), lambda c: 5 + square(c))
>>> foo(-2)
9
```
1.7 Draw the environment diagram that results from running the code.

```python
def dream1(f):
    kick = lambda x: mind()
    def dream2(secret):
        mind = f(secret)
        kick(2)
    return dream2

inception = lambda secret: lambda: secret
real = dream1(inception)(42)
```

https://goo.gl/TbZ1ql

1.8 Fill in the blanks (without using any numbers in the first blank) such that the entire expression evaluates to 9.

```python
(lambda x: lambda y: @lambda: y(x)@)(@3@)(lambda z: z*z)
```

1.9 Draw the environment diagram that results from running the code.

```python
apple = 4
def orange(apple):
    apple = 5
def plum(x):
        return lambda plum: plum * 2
    return plum

orange(apple)("hiii")(4)
```

https://goo.gl/pZ0RLR