# Environment Diagrams & Higher-Order Functions

## Computer Science Mentors 61A

September 9 – September 13, 2024

## 1 Environment Diagrams

1. Give the environment diagram and console output that result from running the following code.

```
def swap(x, y):
    x, y = y, x
    return print("Swapped!", x, y)

x, y = 60, 1
a = swap(x, y)
swap(a, y)
```

Created by Jim Fang, Bryan Adolfo Ventura Benitez. Worksheet feedback: links.jimfang.dev/feedback

2. Draw the environment diagram that results from running the following code.

```
def funny(joke):
    hoax = joke + 1
    return funny(hoax)

def sad(joke):
    hoax = joke - 1
    return hoax + hoax

funny, sad = sad, funny
result = funny(sad(2))
```

## 2 Higher-Order functions

1. Give the environment diagram and console output that result from running the following code.

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

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

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

```
(lambda x: lambda y: ____________________)(______)(lambda z: z*z)()
```

3. Write a function, `whole_sum`, which takes in an integer, `n`. It returns another function which takes in an integer, and returns `True` if the digits of that integer sum to `n` and `False` otherwise.

```
def whole_sum(n):
    """
    >>> whole_sum(21)(777)
    True
    >>> whole_sum(142)(10010101010)
    False
    """
    def check(x):

        ____________________

        while _______________:

            last = ____________________

            ____________________________

            ____________________________

        return ____________________

    return ____________________
```

4. Write a higher-order function that passes the following doctests.

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

```
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
    """
```

5. What would Python display?

   (a) `> (lambda x: x(x))(lambda y: 4)`

   (b) `> (lambda x, y: y(x))(mul, lambda a: a(3, 5))`