

HIGHER-ORDER FUNCTIONS AND SEQUENCES

CS 61A GROUP MENTORING

June 27, 2018

1 Higher-Order Functions

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

2. Draw the environment diagram that results from running the 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))
```

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

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

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

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?

```
>>> foo = mystery(lambda a, b: a(b), lambda c: 5 + square(c))  
>>> foo(-2)
```

2 Sequences

6. Draw box-and-pointer diagrams for the following:

```
>>> a = [1, 2, 3]
>>> a
```

```
>>> a[2]
```

```
>>> b = a
>>> a = a + [4, 5]
>>> a
```

```
>>> b
```

```
>>> c = a
>>> a = [4, 5]
>>> a
```

```
>>> c
```

```
>>> d = c[0:2]
>>> c[0] = 9
>>> d
```

7. Write a function `duplicate_list`, which takes in a list of positive integers and returns a new list with each element x in the original list duplicated x times.

```
def duplicate_list(lst):
    """
    >>> duplicate_list([1, 2, 3])
    [1, 2, 2, 3, 3, 3]
    >>> duplicate_list([5])
    [5, 5, 5, 5, 5]
    """
```

```
    _____
for _____:
        for _____:
            _____
    _____
```