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?

   Solution: 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.

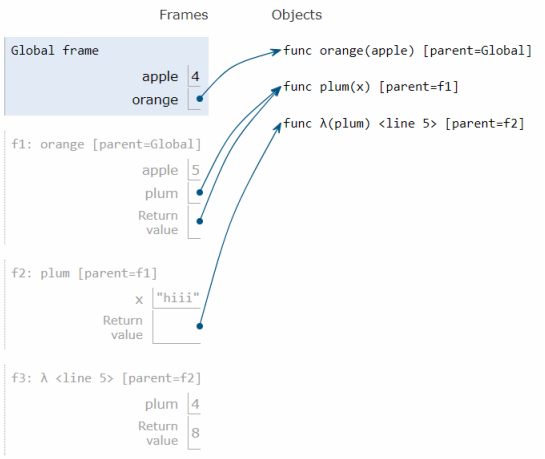
2. Draw the environment diagram that results from running the code.

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

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

   Solution: https://goo.gl/i2yiQF

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)
```



Solution:
https://goo.gl/pZ0RLR

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
    """
```

   Solution:

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

   Challenge solution:

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

5. What would Python display?

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

   Solution:

```
9
```

2 Sequences

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

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

Solution:
```
[1, 2, 3]
```

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

Solution: 3

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

Solution:
```
[1, 2, 3, 4, 5]
```

```
>>> b
```

Solution:
```
[1, 2, 3]
```

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

Solution:
```
[4, 5]
```

```
>>> c
```

Solution:
```
[1, 2, 3, 4, 5]
```

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

Solution:
```
[1, 2]
```

Solution: Box and pointer diagram in Python Tutor.

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 ____________________________________:

            ____________________________________

    ____________________________________
```

Solution:

```
new_list = []
for x in lst:
    for i in range(x):
        new_list = new_list + [x]
return new_list
```