MUTATION AND NONLOCAL

COMPUTER SCIENCE MENTORS CS 61A

February 26 to February 28, 2018

1 Mutation

1. Draw the box-and-pointer diagram.

```
>>> corgi = [3, 15, 18, 7, 9]
>>> husky = [8, 21, 19, 11, 25]
>>> poodle = corgi.pop()
>>> corgi += husky[-3:]
```

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

```
a = [1, 2, [3]]
def mystery(s, t):
    s.pop(1)
    return t.append(s)
b = a
a += [b[0]]
a = mystery(b, a[1:])
```

3. Given some list `lst`, possibly a deep list, mutate `lst` to have the accumulated sum of all elements so far in the list. If there is a nested list, mutate it to similarly reflect the accumulated sum of all elements so far in the nested list. Return the total sum of the orignal `lst`.

   Hint: The **`isinstance`** function returns True for **`isinstance`**`(l, list)` if `l` is a list and False otherwise.

```
def accumulate(lst):
    """
    >>> l = [1, 5, 13, 4]
    >>> accumulate(l)
    23
    >>> l
    [1, 6, 19, 23]
    >>> deep_l = [3, 7, [2, 5, 6], 9]
    >>> accumulate(deep_l)
    32
    >>> deep_l
    [3, 10, [2, 7, 13], 32]
    """
    ____________________________________________
    for ____________________________________________:
        ____________________________________________
        if isinstance(____________________, list):
            inside = ____________________________
            ____________________________________
        else:
            ____________________________________
            ____________________________________
    ____________________________________________
```

2 Nonlocality

1. **Nonlocal Kale**
   Draw the environment diagram for the following code.

```
eggplant = 8
def vegetable(kale):
    def eggplant(spinach):
        nonlocal eggplant, kale
        kale = 9
        eggplant = spinach
        return eggplant + kale
    eggplant(kale)
    return eggplant

spinach = vegetable(10)
```

2. **Pingpong again...**
   Time for some more ping-pong! Remember, the ping-pong sequence counts up starting from 1 and is always either counting up or counting down. At element k, the direction switches if k is a multiple of 7 or contains the digit 7.

   The first 20 elements of the ping-pong sequence are listed below, with direction swaps marked using brackets at the 7th, 14th, and 17th elements
   `1 2 3 4 5 6 [7] 6 5 4 3 2 1 [0] 1 2 [3] 2 1 0`

   Implement a function `make_pingpong_tracker` that returns the next value in the pingpong sequence each time it is called. You may use assignment statements.

```
def has_seven(k): # Use this function for your answer below
    if k % 10 == 7:
        return True
    elif k < 10:
        return False
    else:
        return has_seven(k // 10)

def make_pingpong_tracker():
    """ Returns a function that returns the next value in the
    pingpong sequence each time it is called.
    >>> output = []
    >>> x = make_pingpong_tracker()
    >>> for _ in range(9):
    ...     output += [x()]
    >>> output
    [1, 2, 3, 4, 5, 6, 7, 6, 5]
    """
    index, current, add = 1, 0, True
    def pingpong_tracker():
        _____________________________
        if add:
            ___________________________
        else:
            ___________________________
        if ___________________________:
            add = not add
        _____________________________
        _____________________________

    return pingpong_tracker
```