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:]
```

Solution: https://goo.gl/NbK9YF

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:])
```

Solution: https://goo.gl/s2XKiG

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:
            ____________________________________
            ____________________________________
    ____________________________________________
```

Solution:

```
    sum_so_far = 0
    for i in range(len(lst)):
        item = lst[i]
        if isinstance(item, list):
            inside = accumulate(item)
            sum_so_far += inside
        else:
            sum_so_far += item
            lst[i] = sum_so_far
    return sum_so_far
```

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)
```

Solution: `https://goo.gl/2bmMk9`

2. **Pingpong again...**

   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)
```

Solution:

```
def make_pingpong_tracker():
    index, current, add = 1, 0, True
    def pingpong_tracker():
        nonlocal index, current, add
        if add:
            current = current + 1
        else:
            current = current - 1
        if has_seven(index) or index % 7 == 0:
            add = not add
        index += 1
        return current
    return pingpong_tracker
```