LISTS AND TREES

COMPUTER SCIENCE MENTORS 61A

February 19 to February 21, 2018

1 Lists

1. 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
```

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

```
def reverse(lst):
    if len(lst) <= 1:
        return lst
    return reverse(lst[1:]) + [lst[0]]

lst = [1, [2, 3], 4]
rev = reverse(lst)
```

3. Write a function that takes in a list `nums` and returns a new list with only the primes from `nums`. Assume that `is_prime(n)` is defined. You may use a `while` loop, a `for` loop, or a list comprehension.

```
def all_primes(nums):
```

4. Write a function that takes in a list of positive integers and outputs a list of lists where the i-th list contains the integers from 0 up to, but not including, the i-th element of the input list.

```
def list_of_lists(lst):
"""
>>> list_of_lists([1, 2, 3])
[[0], [0, 1], [0, 1, 2])

>>>list_of_lists([1])
[[0]]

>>>list_of_lists([])
[]
"""
```

2 Trees

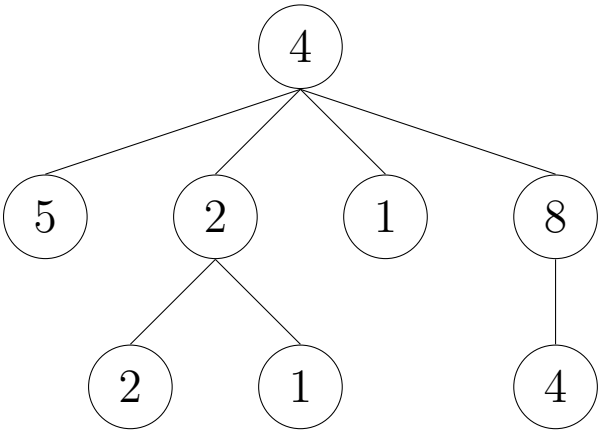
Things to remember:

```
def tree(label, branches=[]):
    return [label] + [branches]

def label(tree):
    return tree[0]

def branches(tree):
    return tree[1:] #returns a list of branches
```

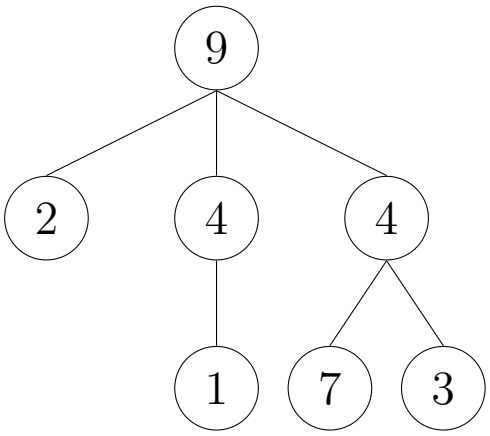
As shown above, the tree constructor takes in a label and a list of branches (which are themselves trees).

```
tree(4,
    [tree(5, []),
     tree(2,
        [tree(2, []),
         tree(1, [])]),
     tree(1, []),
     tree(8,
        [tree(4, [])])])
```

The above expression constructs a tree that looks like this:



1. Construct the following tree and save it to the variable `t`.



2. What would this output? If the output is a tree, write the expression that would create that tree (i.e. `tree(..., ...)`)

```
>>> label(t)

>>> branches(t)[2]

>>> branches(branches(t)[2])[0]
```

3. Write the Python expression to return the integer 2 from t.

4. Write the function `sum_of_nodes` which takes in a tree and outputs the sum of all the elements in the tree.

```
def sum_of_nodes(t):
    """
    >>> t = tree(...) # Tree from question 2.
    >>> sum_of_nodes(t) # 9 + 2 + 4 + 4 + 1 + 7 + 3 = 30
    30
    """
```