

LISTS AND TREES

COMPUTER SCIENCE MENTORS CS 61A

February 19 to February 21, 2018

1 Lists

1. What would Python display? If an error occurs, write "Error". If a function is displayed, write "Function". If nothing is returned, write "Nothing". Also, draw box-and-pointer diagram for each list created.

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

```
>>> a[1]
```

```
>>> a.append([3, 4])
```

```
>>> a
```

```
>>> b = a[1:]
```

```
>>> a[1] = 5
```

```
>>> a[2][0] = 6
```

```
>>> b
```

```
>>> a.extend([7])
```

```
>>> a += [8]
```

```
>>> a
```

```
>>> a += 9
```

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):
```

2 Trees

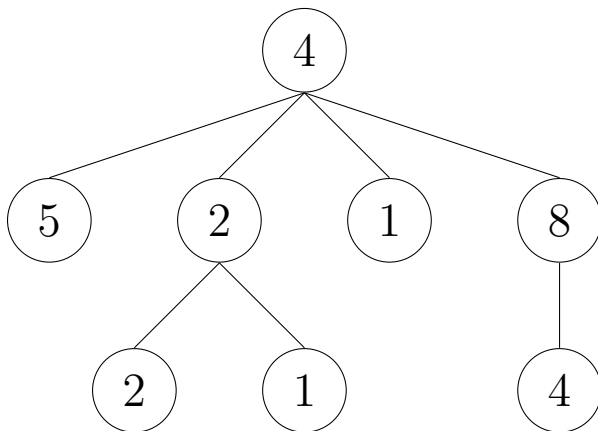
Things to remember:

```
def tree(label, branches=[]):  
    return [label] + list(branches)  
  
def label(tree):  
    return tree[0]  
  
def branches(tree):  
    return tree[1:] #returns a list of branches
```

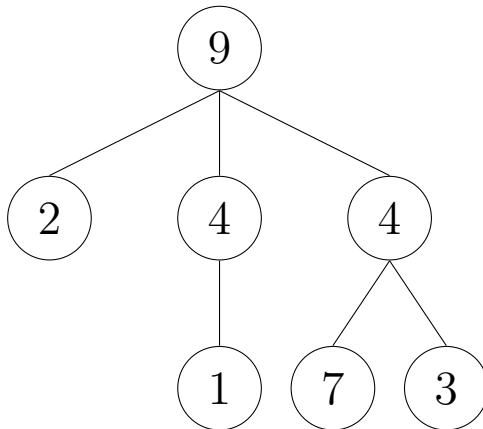
As shown above, the tree constructor takes in a label and a list of branches (which are themselves trees).

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

This creates a tree that looks like this:



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



2. What do the following expressions evaluate to? If the expressions evaluates to a tree, format your answer as `tree(... , ...)`. (Note that the python interpreter wouldn't display trees like this. We ask you to do this in order to think about trees as an ADT instead of worrying about their implementation.)

```
>>> label(t)
```

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

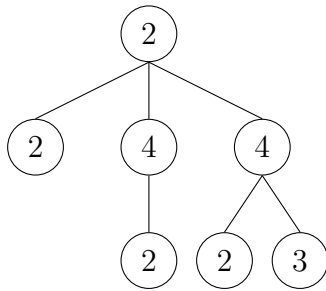
```
>>> branches(branches(t)[2])[0]
```

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.

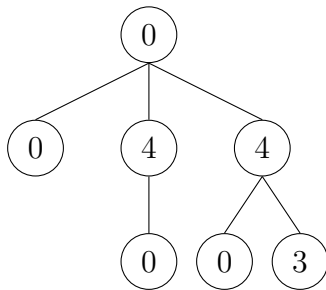
```
def sum_of_nodes(t):
    """
    >>> t = tree(...) # Tree from question 2.
    >>> sum_of_nodes(t) # 9 + 2 + 4 + 4 + 1 + 7 + 3 = 30
    30
    """
```

5. Write a function, `replace_x` that takes in a tree, `t`, and returns a new tree with all labels `x` replaced with 0.

For example, if we called `replace_x(t, 2)` on the following tree:



We would expect it to return



```
def replace_x(t, x):
```