CS 61A
Summer 2017

Linked Lists & Trees

Mentoring 6: July 10, 2017

1 Linked Lists

```
empty = 'X'

def link(first, rest=empty):
    return [first, rest]

def first(s):
    return s[0]

def rest(s):
    return s[1]
```

1.1 What would Python display?

```
s = link(1, link(2, link(3)))
```

(a) `first(s)`

(b) `rest(s)`

(c) `rest(first(s))`

(d) `first(rest(s))`

(e) `rest(rest(s))`

(f) `first(rest(rest(s)))`

1.2 Define the function, `get_item`, which returns the value at index `i` in the linked list, `s`. If the index is greater than the length of the list, return `None`.

```
def get_item(s, i):
    """
    >>> link1 = link(1, empty)
    >>> link21 = link(2, link1)
    >>> link421 = link(4, link21)
    >>> get(link421, 0)
    4
    >>> get(link421, 2)
    1
    >>> get(link421, 999) # returns None
    """
```

1.3 Implement `every_other`, which returns a list containing every other element starting from the *second*.

```
def every_other(s):
    """
    >>> s = link(1, link(2, link(3, link(4, link(5, empty)))))
    >>> print_link(s)
    <1 2 3 4 5>
    >>> print_link(every_other(s))
    <2 4>
    """
```

1.4 Implement `merge`, which takes in two sorted linked lists and returns a sorted linked list that contains all the elements of both.

```
def merge(lst1, lst2):
    """
    >>> l1 = link(2, link(2, link(5, empty)))
    >>> l2 = link(1, link(5, link(6, empty)))
    >>> lst = merge(l1, l2)
    >>> print_link(lst):
    <1 2 2 5 5 6>
    """
```

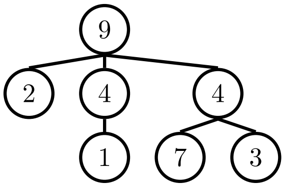
2 Trees

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

def root(tree):
    return tree[0]

def branches(tree):
    return tree[1:]
```

2.1 Draw the tree that is created by the expression to the right:

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

2.2 Assign the name, `t`, to the tree to the right.



2.3 What would Python display?

(a) `root(t)`

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

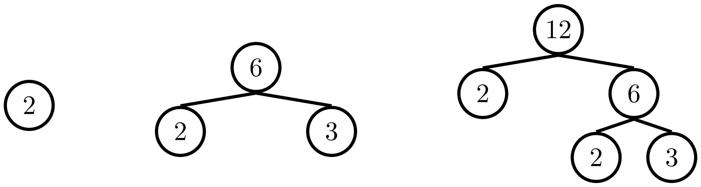
(c) `branches(branches(t)[2])[0]`

2.4 Write the Python expression to return the integer 2 from `t`.

2.5 Define the function `tree_sum` which takes in a tree and outputs the sum of all the values in the tree.

```
def tree_sum(t):
    """
    >>> t = tree(...)   # Example from earlier
    >>> tree_sum(t) # 9 + 2 + 4 + 4 + 1 + 7 + 3 = 30
    30
    """
```

2.6 Define the function `factor_tree` which returns a *factor tree.* Recall that in a factor tree, multiplying the leaves together is the prime factorization of the root, n.



```
def factor_tree(n):
```

2.7 Define the function `count` which counts the number of instances of a `value` in the given tree.

```
def count(t, value):
```