Lab 9: Mutable Trees **lab09.zip (lab09.zip)**

Due by 11:59pm on Wednesday, October 25.

Starter Files

Download lab09.zip (lab09.zip). Inside the archive, you will find starter files for the questions in this lab, along with a copy of the Ok (ok) autograder.

Topics

Consult this section if you need a refresher on the material for this lab. It's okay to skip directly to the questions and refer back here should you get stuck.

Mutable Trees

Required Questions

Getting Started Videos

Mutable Trees

Q1: WWPD: Trees

Read over the `Tree` class in `lab09.py`. Make sure you understand the doctests.

> Use Ok to test your knowledge with the following "What Would Python Display?" questions:
>
> ```
> python3 ok -q trees-wwpd -u
> ```
>
> Enter `Function` if you believe the answer is `<function ...>`, `Error` if it errors, and `Nothing` if nothing is displayed. Recall that `Tree` instances will be displayed the same way they are constructed.

```
>>> from lab09 import *
>>> t = Tree(1, Tree(2))
------

>>> t = Tree(1, [Tree(2)])
>>> t.label
------

>>> t.branches[0]
------

>>> t.branches[0].label
------

>>> t.label = t.branches[0].label
>>> t
------

>>> t.branches.append(Tree(4, [Tree(8)]))
>>> len(t.branches)
------

>>> t.branches[0]
------

>>> t.branches[1]
------
```

Q2: Cumulative Mul

Write a function `cumulative_mul` that mutates the Tree `t` so that each node's label becomes the product of its label and all labels in the subtrees rooted at the node.

> **Hint**: Consider carefully whether the mutation of the tree should happen before or after processing the subtrees.

```
def cumulative_mul(t):
    """Mutates t so that each node's label becomes the product of all labels in
    the corresponding subtree rooted at t.

    >>> t = Tree(1, [Tree(3, [Tree(5)]), Tree(7)])
    >>> cumulative_mul(t)
    >>> t
    Tree(105, [Tree(15, [Tree(5)]), Tree(7)])
    >>> otherTree = Tree(2, [Tree(1, [Tree(3), Tree(4), Tree(5)]), Tree(6, [Tree(7)])])
    >>> cumulative_mul(otherTree)
    >>> otherTree
    Tree(5040, [Tree(60, [Tree(3), Tree(4), Tree(5)]), Tree(42, [Tree(7)])])
    """
    "*** YOUR CODE HERE ***"
```

Use Ok to test your code:

```
python3 ok -q cumulative_mul
```

Q3: Prune Small

Complete the function `prune_small` that takes in a `Tree t` and a number `n` and prunes `t` mutatively. If `t` or any of its branches have more than `n` branches, the `n` branches with the smallest labels should be kept and any other branches should be *pruned*, or removed, from the tree.

> *Hint*: The `max` function takes in an `iterable` as well as an optional `key` argument (which takes in a one-argument function). For example, `max([-7, 2, -1], key = abs)` would return `-7` since `abs(-7)` is greater than `abs(2)` and `abs(-1)`.

```
def prune_small(t, n):
    """Prune the tree mutatively, keeping only the n branches
    of each node with the smallest labels.

    >>> t1 = Tree(6)
    >>> prune_small(t1, 2)
    >>> t1
    Tree(6)
    >>> t2 = Tree(6, [Tree(3), Tree(4)])
    >>> prune_small(t2, 1)
    >>> t2
    Tree(6, [Tree(3)])
    >>> t3 = Tree(6, [Tree(1), Tree(3, [Tree(1), Tree(2), Tree(3)]), Tree(5, [Tree(3), Tre
    >>> prune_small(t3, 2)
    >>> t3
    Tree(6, [Tree(1), Tree(3, [Tree(1), Tree(2)])])
    """
    while _____________________________:
        largest = max(_________________, key=____________________)
        _________________________
    for __ in _____________:
        ___________________
```

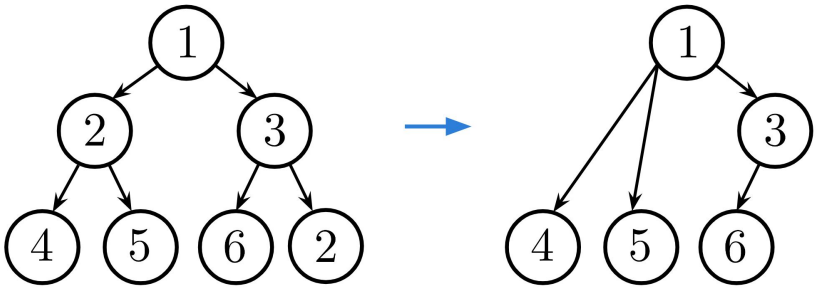
Use Ok to test your code:

```
python3 ok -q prune_small
```

Optional Questions

Q4: Delete

Implement `delete`, which takes a Tree `t` and deletes any occurrence of `x` within it. The order of the branches must be preserved.



> *Important*: When a non-leaf node is deleted, the deleted node's children should be attached to the deleted node's parent. You may assume that the root of the tree will never be deleted.

```
def delete(t, x):
    """
    Delete any occurrence of the 'x' within Tree 't'. When a non-leaf
    node is deleted, the deleted node's children should be attached to
    its parent. The order of the branches must be preserved.

    Assume that the root will never be deleted.

    >>> t = Tree(3, [Tree(2, [Tree(2), Tree(2)]), Tree(2), Tree(2, [Tree(2, [Tree(2), Tree
    >>> delete(t, 2)
    >>> t
    Tree(3)
    >>> t = Tree(1, [Tree(2, [Tree(4, [Tree(2)]), Tree(5)]), Tree(3, [Tree(6), Tree(2)]),
    >>> delete(t, 2)
    >>> t
    Tree(1, [Tree(4), Tree(5), Tree(3, [Tree(6)]), Tree(4)])
    >>> t = Tree(1, [Tree(2, [Tree(4), Tree(5)]), Tree(3, [Tree(6), Tree(2)]), Tree(2, [Tr
    >>> delete(t, 2)
    >>> t
    Tree(1, [Tree(4), Tree(5), Tree(3, [Tree(6)]), Tree(6), Tree(7), Tree(8), Tree(4)])
    """
    new_branches = []
    for _________ in _________________:
        ______________________
        if b.label == x:
            ___________________________________
        else:
            ___________________________________
    t.branches = ____________________
```

Use Ok to test your code:

```
python3 ok -q delete
```