8. (8 points) Leaf It To Me

Write a function `max_path` which takes in a tree `t` with positive labels and a number `k`, and returns the path with length at most `k` for which the sum of the labels in the path is greatest. If there are multiple paths with the greatest sum, return the leftmost one.

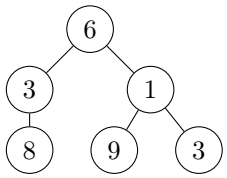
The path does *not* have to start at the root of the tree - the path can contain any top-to-bottom sequence of nodes.

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

def label(t):
    return t[0]

def is_leaf(t):
    return not branches(t)

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

The tree data abstraction is provided here for your reference. Do not violate the abstraction barrier!

```
def max_path(t, k):
    """ Return a list of the labels on any path in tree t of length at most k with the greatest sum

    >>> t1 = tree(6, [tree(3, [tree(8)]), tree(1, [tree(9), tree(3)])])
    >>> max_path(t1, 3)
    [6, 3, 8]
    >>> max_path(t1, 2)
    [3, 8]
    >>> t2 = tree(5, [t1, tree(7)])
    >>> max_path(t2, 1)
    [9]
    >>> max_path(t2, 2)
    [5, 7]
    >>> max_path(t2, 3)
    [6, 3, 8]
    """
    def helper(t, k, on_path):

        if ____________________________________________________________________:

            return []

        elif ____________________________________________________________________:

            return [label(t)]

        a =  ____________________________________________________

        if ____________________________________________________________________:

            return ____________(____________________, key = ________________________)

        else:

            b = ____________________________________________________

            return _____________(____________________, key = ________________________)

    return helper(t, k, False)
```

t1



t2