CSC148 - More Practice with Trees

1. Consider the following `Tree` method:

```
def leaves(self) -> List:
    """Return a list of all of the leaf items in the tree.
    """
```

   (a) What should `leaves` return when called on an empty tree?

   (b) What should `leaves` return when called on a tree of size 1?

   (c) Suppose we have a tree `t` with three subtrees `t1`, `t2`, and `t3`, with the following partial tracing table:

tree	tree.leaves()
t1	[1, 3, 5]
t2	[-4]
t3	[10, 20]

   Using this information, determine what `t.leaves()` should return.

   (d) Finally, implement this method in the space below.

2. Now consider this method:

```
def average(self) -> float:
    """Return the average of all the values in this tree.

    Return 0.0 if this tree is empty.

    Precondition: this is a tree of numbers.
```

(a) Suppose we have a tree `t` with three subtrees `t1`, `t2`, and `t3`, and consider a hypothetical partial tracing table:

tree	tree.average()
t1	3.0
t2	-4.5
t3	0.0

Can we determine what `t.average()` should return from this information? Why or why not?

(b) To get around this problem, we will not make `Tree.average` directly recursive. First, suppose we have a tree `t`. What are the *two* pieces of information we need to calculate the average of the numbers in this tree?

(c) In the space below, implement a *recursive helper method* that recursively computes the two values you identified in the previous part. Note that this helper should return a **tuple** containing the two values.

(d) Finally, show how to implement `Tree.average` using your recursive helper. This should be extremely short!

```
def average(self) -> float:
```