Write recursive contains method

first...

Read over the **_init_** method for class **Tree**:

```
class Tree:
    """
    A bare-bones Tree ADT that identifies the root with the entire tree.
    """

    def __init__(self, value=None, children=None):
        """
        Create Tree self with content value and 0 or more children

        @param Tree self: this tree
        @param object value: value contained in this tree
        @param list[Tree|None] children: possibly-empty list of children
        @rtype: None
        """
        self.value = value
        # copy children if not None
        self.children = children.copy() if children else []
```

next...

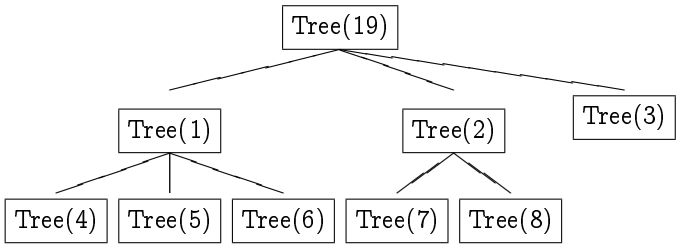
Now, read the header and docstring for the method **contains**, and then answer the questions that follow it.

```
    def __contains__(self, v):
        """
        Return whether Tree self contains v.

        @param Tree self: this tree
        @param object v: value to search this tree for

        >>> t = Tree(17)
        >>> t.__contains__(17)
        True
        >>> t = descendants_from_list(Tree(19), [1, 2, 3, 4, 5, 6, 7], 3)
        >>> t.__contains__(5)
        True
        >>> t.__contains__(18)
        False
        """
```

then...

1. One of the examples in **contains** docstring is simple enough not to require recursion. Write an **if...** expression that checks for this case, and then returns the correct thing. Include an **else...** for when the tree is **less** easy to deal with.

2. Below is a picture of a larger **Tree** with several levels. Consider a function call **contains(t, 18)**, supposing **t** refers to the root of the tree. Are there smaller trees for which it would be helpful to know whether they contain 18? Which smaller trees are they? Write an example of a function call **contains(??, 18)** on one of these smaller trees. Rather than **??**, You can access these trees through the variable **t**.



3. Suppose the call in the previous step gives you the correct answer according to the docstring: it returns whether or not the smaller tree contains 18. How will you combine the solutions for all the smaller instances to get a solution for **Tree t** itself? Write code to return the correct thing. Put this code in the **else...** expression that you created in the first step.