class Tree:
    """A recursive tree data structure."""

    # === Private Attributes ===
    # @type _root: object | None
    # The item stored at the tree's root, or None if the tree is empty.
    # @type _subtrees: list[Tree]
    # A list of all subtrees of the tree.

    # === Representation Invariants ===
    # - If self._root is None then self._subtrees is empty.
    # - This setting of attributes represents an empty tree.
    # - self._subtrees doesn't contain any empty trees

def __init__(self, root):
    """Initialize a new Tree with the given root value.

    If <root> is None, the tree is empty.
    A new tree always has no subtrees.
"

    @type self: Tree
    @type root: object | None
    @rtype: None

def add_subtrees(self, new_trees):
    """Add the trees in <new_trees> as subtrees of this tree.
"

    @type self: Tree
    @type new_trees: list[Tree]
    @rtype: None

1. Write a Python expression to create an empty tree.

2. Write a Python expression to create a tree containing just the single item ‘hello’.

3. Write a sequence of Python expressions which create a tree with a root value of 1, whose two children are 10 and 20. You may use temporary variables in your answer, and you should call add_subtrees.
Here is a diagram of a fairly large tree.

4. What is the size of this tree?

5. Draw each subtree of this tree. Underneath each one, write its size.

6. In English, explain the relationship between the size of a tree and the size of its subtrees.

7. Implement the method `__len__` for the Tree class, which returns the size of the tree.

```python
def __len__(self):
    """Return the size of this tree.""
```