Salary Inversion

Two requirements:

- Your solution to this problem must use a recursive function.
- Your solution must not use global variables.

Solutions that violate either of these requirements will receive no credit.

A company is trying to determine if there are workers who make more than their bosses. A *boss* has one or more direct reports. The *CEO* of the company has zero or more direct reports, but does not have a boss. A *worker* has no direct reports. The *management chain* for a worker includes their boss, their boss's boss, etc up to the CEO of the company.

Each employee has a salary and is denoted as Employee N, where N is a unique identifier for the employee. For example, lets assume a company has employees with the following salaries:

- Employee 1 = $6000
- Employee 2 = $5000
- Employee 3 = $3000
- Employee 4 = $2500
- Employee 5 = $5500
- Employee 6 = $10000
- Employee 7 = $9000
- Employee 8 = $1000

Each employee reports to a single direct boss, with the exception of Employee 1 who is always the CEO at the company. Using the list of employees above, the company's direct reporting structure is as follows:

- (Boss: Employee 1, Direct Reports: (Employee 2, Employee 3, Employee 8))
- (Boss: Employee 2, Direct Reports: (Employee 4, Employee 5))
- (Boss: Employee 3, Direct Reports: (Employee 6))
- (Boss: Employee 6, Direct Reports: (Employee 7))

Your goal is to determine the number of workers who have at least one person in their management chain who makes less than they do.

Given this reporting structure, Employees 4, 5, 7, and 8 are workers. Two of whom–5 and 7–make more than at least one boss in their management chain. Employee 5 makes more than their direct boss Employee 2. Employee 7 makes more than Employee 3, their boss's boss, and Employee 1, the CEO.

The reporting structure for the company will be represented using the `Tree` class, which can be found in the file `tree.py`:

```
'''


```

```
Tree class for representing an employee and their subordinates. Used in the
inversion problem.

Please do not modify this file. Your solution should be implemented in the
corresponding problem.py file.
'''

from typing import List

class Tree:
    '''
    Class encapsulating information about an employee and their subordinates.

    Public attributes:
        -   employee_num (int): The employee number of the employee.
        -   salary (int): The salary of the employee.
        -   children (List[Tree]): The subordinates of the employee.

    Public methods:
        -    add_direct_report: Add a new direct report to the tree
        -    print_tree: Print the contents of the tree
    '''
    def __init__(self, name: int, salary: int) -> None:
        '''
        Constructor for Tree class.

        Parameters:
            -   name (str): The name of the employee.
            -   salary (int): The salary of the employee.

        Returns:
            None
        '''
        self.name = name
        self.salary = salary
        self.children: List['Tree'] = []

    def add_direct_report(self, other_tree: 'Tree') -> None:
        '''
        Add a child to this tree corresponding to a subordinate employee of the
        parent tree.

        Parameters:
            -   other_tree (Tree): The tree to add as a child.

        Returns:
```

```
            None
        '''
        self.children.append(other_tree)

    def print_tree(self, indent: int = 0) -> None:
        '''
        Print the contents of this tree.

        Parameters:
            -   indent (int): The number of spaces to indent the tree by.

        Returns:
            None
        '''
        print(" " * indent + f"{self.name} {self.salary}")

        for child in self.children:
            child.print_tree(indent + 4)
```

Your task is to complete the function `solve` in `problem8.py`. Given the root of the tree (the `Tree` node representing the CEO), your function should return the number of workers who make more than at least one boss in their management chain.

Here is the skeleton code for `problem8.py`:

```
'''
Distribution file of the inversion problem
'''

from tree import Tree


def solve(tree: Tree) -> int:
    '''
    Recursive function to find the number of workers who are paid more than
    their boss. Note that a worker is defined as someone who has no
    subordinates.

    Parameters:
        -   tree (Tree): The root of the tree of employees.

    Returns:
        The number of workers who are paid more than their at least one of their
        bosses.
    '''
    # TODO: Implement this function
```

```
pass
```