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.java`:

```
import java.util.List;
```

```
import java.util.ArrayList;

/**
 * Class for representing an employee and the people who report to them.
 *
 * Public attributes:
 *   empNum: the employee's employee number represented as an integer
 *   salary: the employee's salary represented as an integer
 *   children: a list of the employee's direct reports
 *
 * Public methods:
 *   addDirectReport: add a new direct report to the tree
 *   printTree: print the contents of the tree.
 **/
public class Tree {
    public int empNum;
    public int salary;
    public List<Tree> children;

    /**
     * Constructor for Tree
     * Arguments:
     *   empNum: the employee's employee number represented as an integer
     *   salary: the employee's salary represented as an integer
     **/
    public Tree(int empNum, int salary) {
        this.empNum = empNum;
        this.salary = salary;
        this.children = new ArrayList<Tree>();
    }

    /**
     * addDirectReport: add a new direct report
     *
     * Arguments:
     *   t: the new direct report represented as a Tree
     **/
    public void addDirectReport(Tree t) {
        this.children.add(t);
    }

    /**
     * printTree: print the contents of the tree, indented
     *   to illustrate the tree structure
     *
     * Arguments:
```

```
     *   tabs: a string with spaces that tracks the indentation level
     **/
    public void printTree(String tabs) {
        System.out.println(tabs + this.empNum + " " + this.salary);
        for (Tree t : this.children) {
            t.printTree(tabs + "  ");
        }
    }
}
```

Your task is to complete the function `solve` in `Problem8.java`. 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.java`:

```
/* Solution to paths problem.
 * Version w/ integer assignment ids
 */

import java.util.Arrays;
import java.util.Scanner;
import java.util.HashMap;
import java.util.ArrayList;
import java.util.List;
import java.util.Collections;

public class Problem8 {
    /**
     * solve: find the number of Workers (ie. employees with no direct reports)
     *      who make more than at least one of their bosses.
     *
     * t: the root of a Tree representing the company.
     *
     * Returns: the number of workers in the company who make more
     *   than at least one of their bosses.
     **/
    public static int solve(Tree t) {
        // YOUR CODE HERE
        // Return is included to ensure that the skeleton code compiles.
        return -1;
    }
}
```