Programming Part

Problem 4 - Fake Binary Search Tree (100 pts)

Time Limit	:	1 s
Memory Limit	:	1024 MB

Problem Description

Giver, just like you, is a diligent student of the DSA class. He just learned a new data structure: *Binary Search Tree*. Let us recap Binary Search Tree (BST) a bit. A BST is a rooted binary tree that can help you find elements fast. Formally, if a binary tree satisfies the following constraints, then it is a BST.

- Every node of the tree has its own key.
- The key in each node is greater than or equal to any keys stored in its left sub-tree.
- The key in each node is less than or equal to any keys stored in its right sub-tree.

Then, the following algorithm, "binary search", checks whether an element x is in the BST.

Algorithm 2: Lookup of a key in a BST.

Input: A BST rooted at *tree_node* and a *key* to look for

Output: **True** if the *key* is found, **False** otherwise

Function SEARCH*(tree_node, key):*

```
if tree_node == NIL then
    return False
end
if tree_node.key == key then
    return True
end
if tree_node.key > key then
    return SEARCH(tree_node.leftchild, key)
end
else
    return SEARCH(tree_node.rightchild, key)
end
```

After studying this simple algorithm, Giver is super excited. He tries to apply it on general binary trees to see understand more about the algorithm. To his dismay, the algorithm does

not work for general binary trees. Giver is curious about the number of keys that are in the general binary tree which can be found by applying the algorithm. Please help Giver locate those keys and compute the number!

Input

The first line of the input contains only one integer N ($1 \leq N \leq 10^6$), indicating the number of nodes in the given binary tree. The nodes are indexed from 1 to N. In the following N lines, the ith line contains the content of node i, which is represented by three integers w, l, r ($1 \leq w \leq 10^9, l, r \in \{-1\} \cup \{1, 2, \ldots, N\}$), representing the key, the id of left child, and the id of right child. An index of -1 is used to indicate NIL—that is, when there is no child. The keys in each node are unique. It is guaranteed that the input forms a valid binary tree rooted at 1.

Output

Print an integer representing the number of keys (within the binary tree) that can be found by the algorithm (returns **True**).

Subtask 1 (30 pts)

- $N \leq 1000$.

Subtask 2 (70 pts)

- No other constraints.

Sample Input 1

```
5
1 2 4
2 -1 3
5 -1 -1
6 -1 5
7 -1 -1
```

Sample Output 1

```
3
```

Sample Input 2

```
6
2 2 -1
1 -1 3
5 4 6
6 5 -1
7 -1 -1
9 -1 -1
```

Sample Output 2

```
2
```

Sample Input 3

```
7
10 2 7
3 3 4
1 -1 -1
7 5 -1
9 -1 6
5 -1 -1
4 -1 -1
```

Sample Output 3

```
4
```