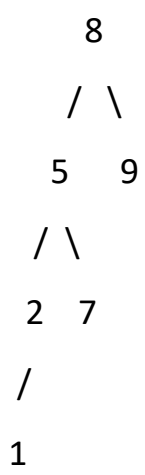
Check whether BST contains Dead End

Given a Binary Search Tree that contains **unique positive integer values greater than 0**. The task is to complete the function **isDeadEnd** which returns **true** if the BST contains a **dead end** else returns **false**. Here **Dead End** means a **leaf** node, at which no other node can be inserted.

Example 1:

Input :

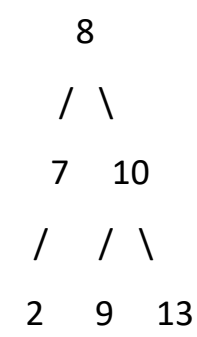


Output :
Yes

Explanation :
Node 1 is a Dead End in the given BST.

Example 2:

Input :



Output :
Yes

Explanation :
Node 9 is a Dead End in the given BST.

Your Task: You don't have to input or print anything. Complete the function **isDeadEnd()** which takes **BST** as input and returns a boolean value.

Expected Time Complexity: O(N), where **N** is the number of nodes in the **BST.**
Expected Space Complexity: O(N)

Constraints:
1 <= N <= 1001
1 <= Value of Nodes <= 10001

Try more examples

```
# function should return true/false or 1/0
class Solution:
    def isDeadEnd(self, root):
        def checkDeadEnd(node, min_val, max_val):
            if not node:
                return False
            if min_val == max_val:
                return True

            return (checkDeadEnd(node.left, min_val,
node.data - 1) or
                    checkDeadEnd(node.right, node.data +
1, max_val))

        return checkDeadEnd(root, 1, 10000)
```