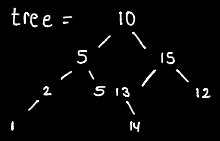
Input: tree =



A potentially invalid BST

Output: true

A boolean representing whether the BST is valid

```
class BST {
  constructor(value) {
    this.value = value;
    this.left = null;
    this.right = null;
  }
}

// O(n) time | O(d) space
function validateBst(tree) {
  return validateBstHelper(tree, -Infinity, Infinity);
}

function validateBstHelper(tree, minValue, maxValue) {
  if (tree === null) return true;
  if (tree.value < minValue || tree.value >= maxValue) return false;
  const isValid = validateBstHelper(tree.left, minValue, tree.value);
  return isValid && validateBstHelper(tree.right, tree.value, maxValue);
}
```

Time: O(n) (where n is the # of nodes) since we touch every node in the tree to check its validity. Every other operation is O(1)

Space: O(d) (where d is the depth of the BST) since we are using frames on the call stack bc of our recursive calls (d is 4 in this case). If only one branch, O(n)