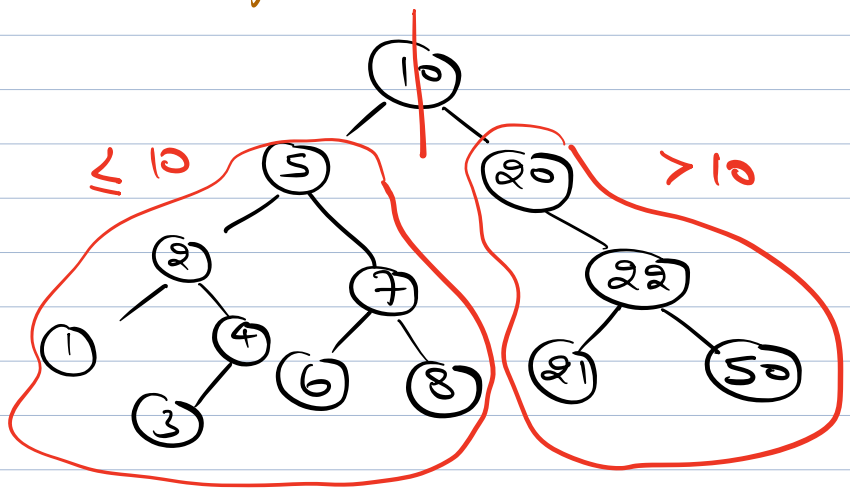
Q. Given a BST. Print the nodes in increasing order.



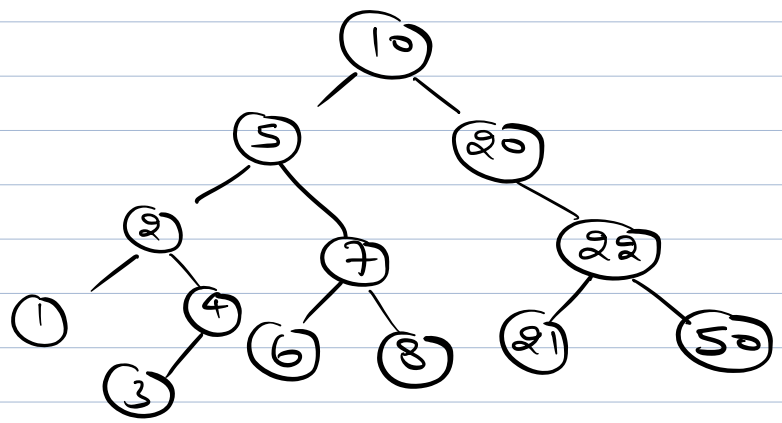
1, 2, 3, 4, 5, 6, 7, 8, 10, 20, 21, 22, 50

← LST → ← RST →

Inorder traversal of a BST is always sorted.

Q. Given a BT.

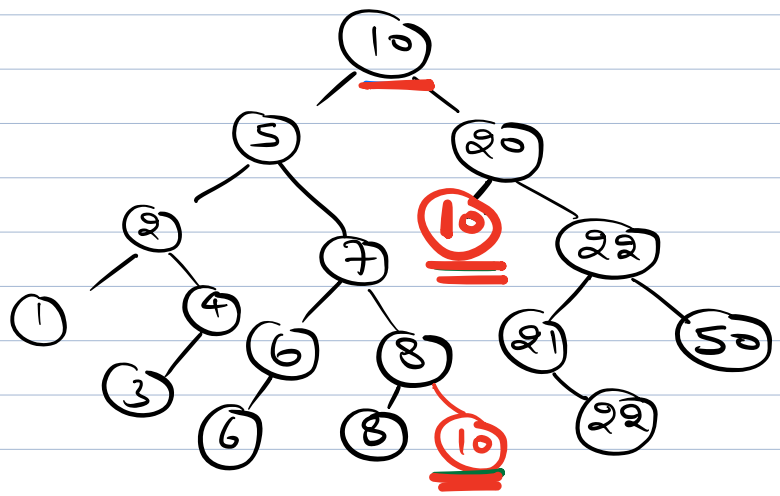
Return True. if it is a BST (No duplic)



1) Inorder

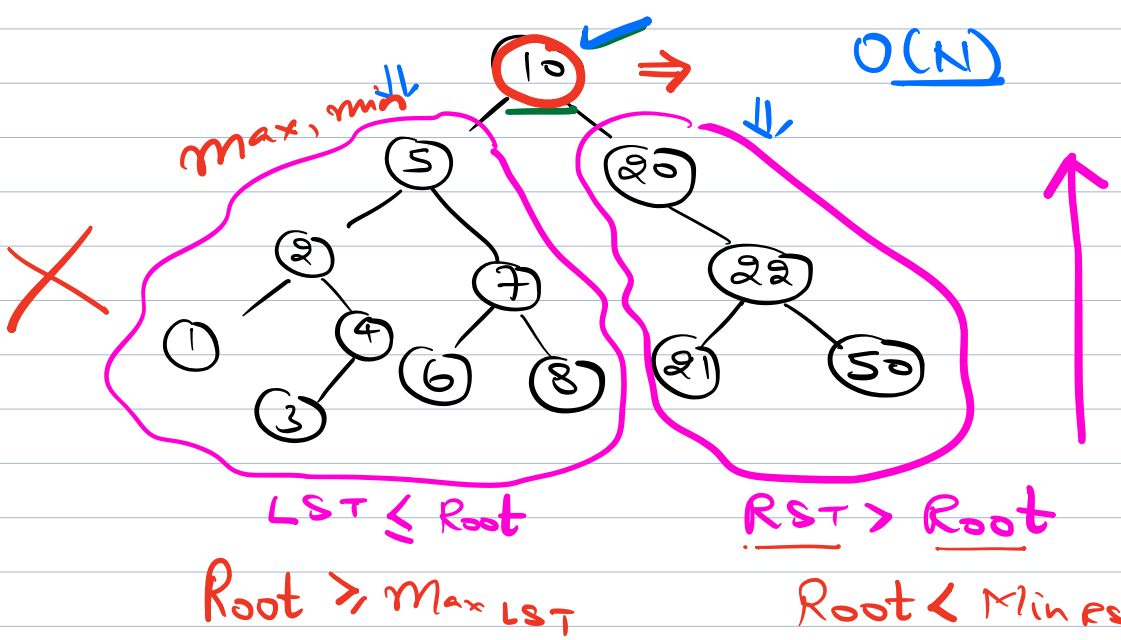
If inorder sorted ⇒ BST

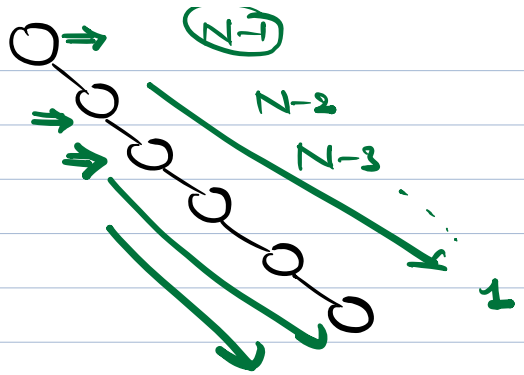
1.1 If duplicates are allowed.



1, 2, 3, 4, 5, 6, 6, 7, 8, 8, 10, 10, 20, 21, 22, 22, 50

2)



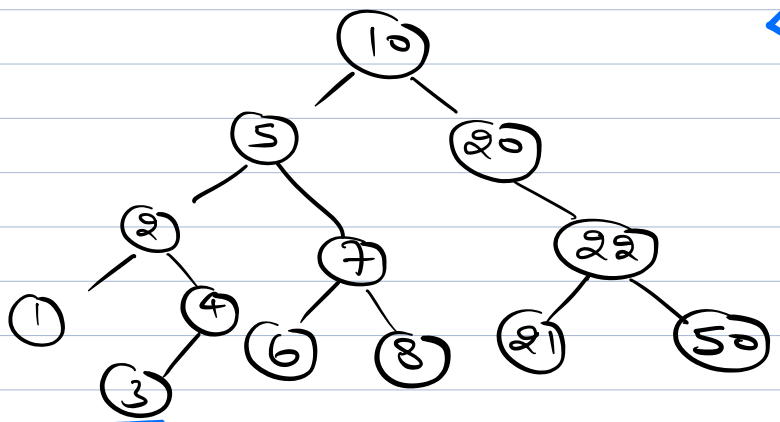


$= O(N^2)$

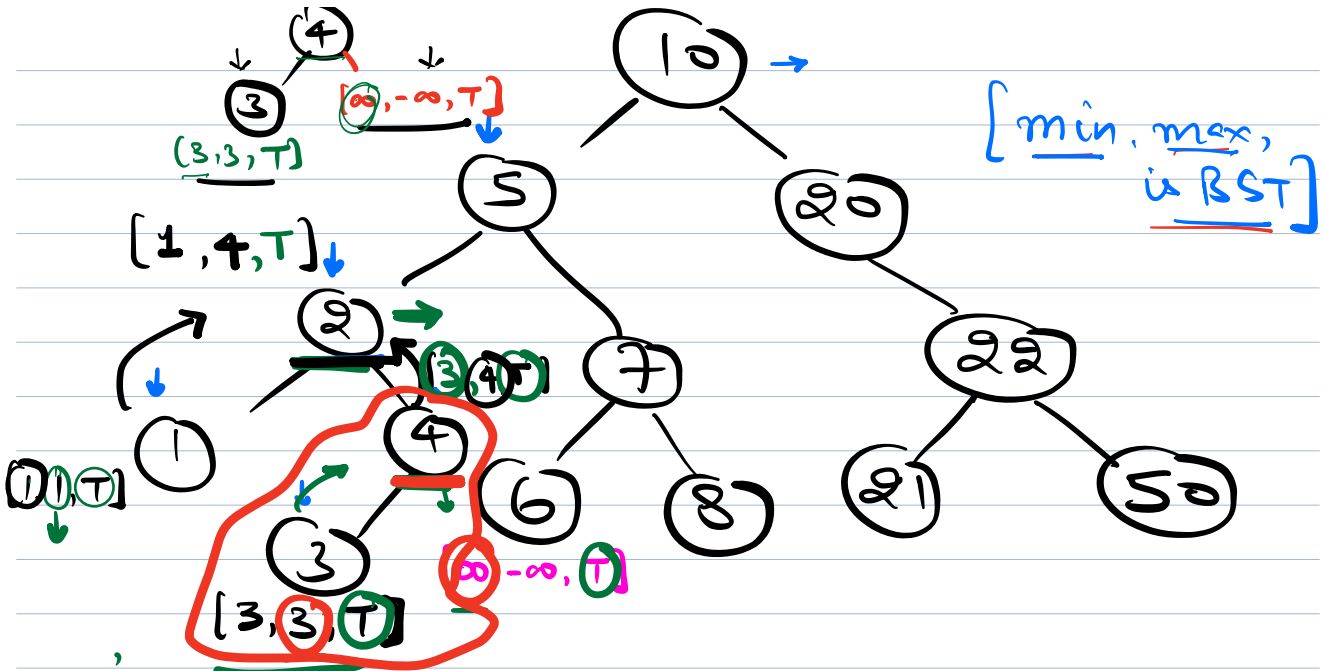
N, L, R ⇒ Pre Order

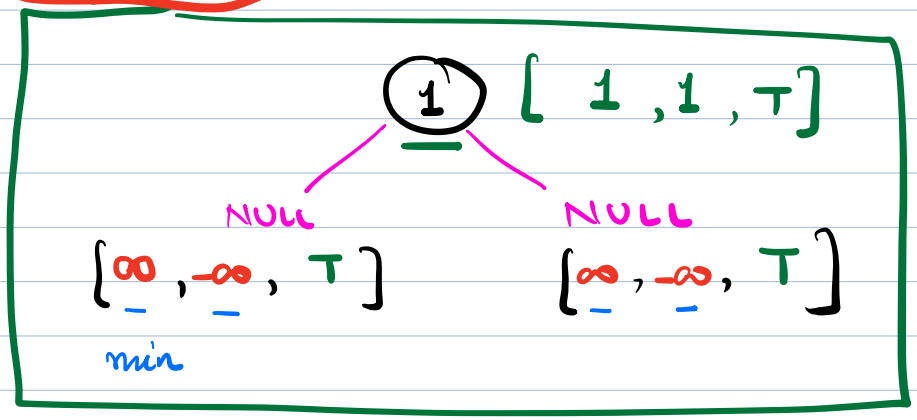
3) Post Order Traversal

<max, min>

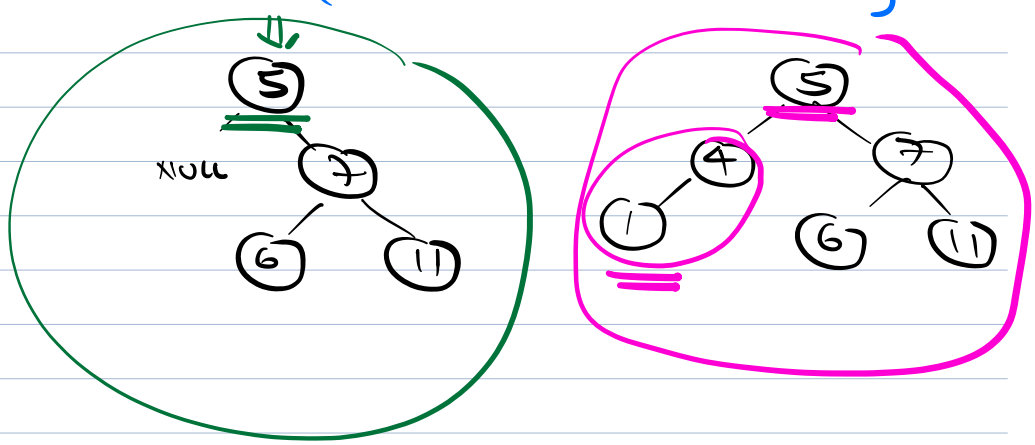



```
class TreeInfo {
    int min;
    int max;
    bool isBST;
};
```





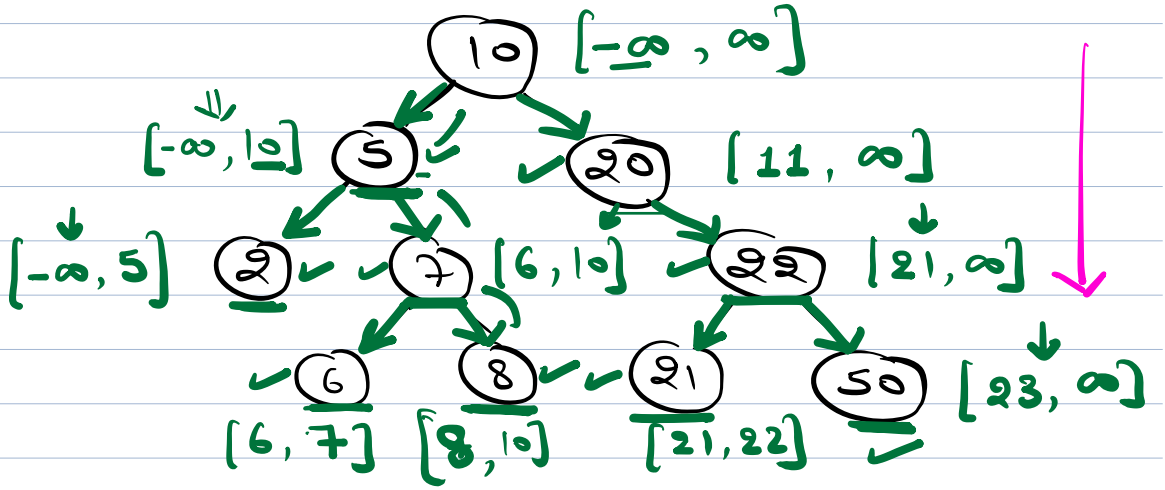
min = min(root.val, min LST)
max = max(root.val, max RST)



```
TreeInfo checkBST(root) {
    if (root == NULL) {
        return new TreeInfo(INT_MAX, INT_MIN, T);
    }

    TreeInfo left = checkBST(root.left);
    TreeInfo right = checkBST(root.right);

    if ((left.isBST == True) && (right.isBST == True)
        && (root.value >= left.max) &&
            (root.value < right.min)) {
        return new TreeInfo(min(root.value,
            left.min), max(root.value, right.max,
            True);
    }

    return new TreeInfo(∞, -∞, False);
}
```

T.C. = $O(N)$

4) Pre Order




```
boolean isBST (root, min, max) {
  if (root == NULL) { return True; }
  if (root.value >= min && root.value
                          < max) {
    boolean left = isBST (root.left,
                   min, root.value);
    boolean right = isBST (root.right,
                   root.value+1, max);
    return (left && right);
  }
  return false;
```