DAY-33

Find kth largest element in Binary Search Tree.

Approach:-

→ The idea is to ~~traverse~~ do reverse inorder traversal of BST. keep a count of nodes visited.

→ The reverse inorder traversal traversed all nodes in decreasing order.
i.e. visit the right node then enter then left and continue traversing the nodes recursively.

→ while doing the traversal, keep track of the count of nodes visited so far.

→ When the count becomed equal to k,
* stop the traversal and return the key.

code part: Implementation :-

```
class Node:
    . . .
    . . .
def kthLargestUtill(root, k, c):
    if root == None or c[0] >= k:
        return
    kthLargestUtill(root.right, k, c)
    c[0] += 1

    if c[0] == k:
        print(root
        return (root.key)
    kthLargestUtill(root.left, k, c)

def kthLargest(root, k):
    c = [0]
    kthLargestUtill(root, k, c)
```

Time Complexity :-> O(h+k)
- first traverses down to the rightmost node which takes O(h) time, then traverses k elements in O(k) time. Hence, O(h+k)

Space Complexity :-> O(1)