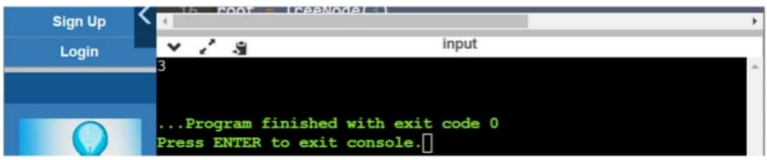
Problem 2: Find the **Maximum Depth** of Binary Tree. Maximum Depth is the **count of nodes of the longest path** from the root node to the leaf node.

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right


def maxDepth(root):
    if root is None:
        return 0

    left_depth = maxDepth(root.left)
    right_depth = maxDepth(root.right)

    return max(left_depth, right_depth) + 1

root = TreeNode(3)
root.left = TreeNode(9)
root.right = TreeNode(20)
root.right.left = TreeNode(15)
root.right.right = TreeNode(7)

depth = maxDepth(root)
print(depth)
```



Problem 6: Given two Binary Tree. Write a program to check if two trees are identical or not.

```
class Node:
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None

def are_identical(root1, root2):
    if root1 is None and root2 is None:
        return True
    if root1 is None or root2 is None:
        return False

    if root1.value != root2.value:
        return False
    return are_identical(root1.left, root2.left) and are_identical(root1.right, root2.right)

tree1 = Node(1)
tree1.left = Node(2)
tree1.right = Node(3)
```

```
tree1.left.left = Node(4)
tree1.left.right = Node(5)

tree2 = Node(1)
tree2.left = Node(2)
tree2.right = Node(3)
tree2.left.left = Node(4)
tree2.left.right = Node(5)

if are_identical(tree1, tree2):
    print("The trees are identical.")
else:
    print("The trees are not identical.")
```

```
tree1 = Node(1)
tree1.left = Node(2)
tree1.right = Node(3)
```

```
tree1.left.left = Node(4)
tree1.left.right = Node(5)

tree2 = Node(1)
tree2.left = Node(2)
tree2.right = Node(3)
tree2.left.left = Node(4)
tree2.left.right = Node(5)

if are_identical(tree1, tree2):
    print("The trees are identical.")
else:
    print("The trees are not identical.")
```