**ASSIGNMENT-24**

Question-1:

Given preorder of a binary tree, calculate its [**depth(or height)**](https://www.geeksforgeeks.org/write-a-c-program-to-find-the-maximum-depth-or-height-of-a-tree/) [starting from depth 0]. The preorder is given as a string with two possible characters.

1. ‘l’ denotes the leaf
2. ‘n’ denotes internal node

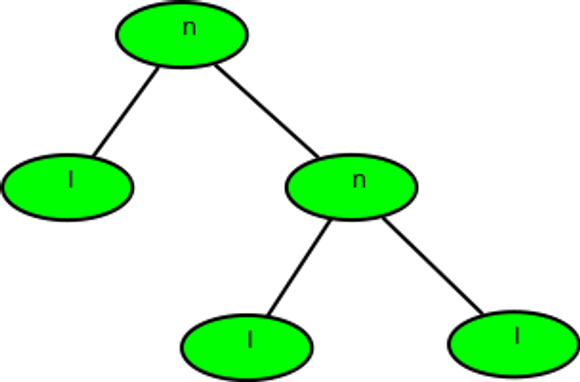
The given tree can be seen as a full binary tree where every node has 0 or two children. The two children of a node can ‘n’ or ‘l’ or mix of both.

**Examples :**

Input : nlnll

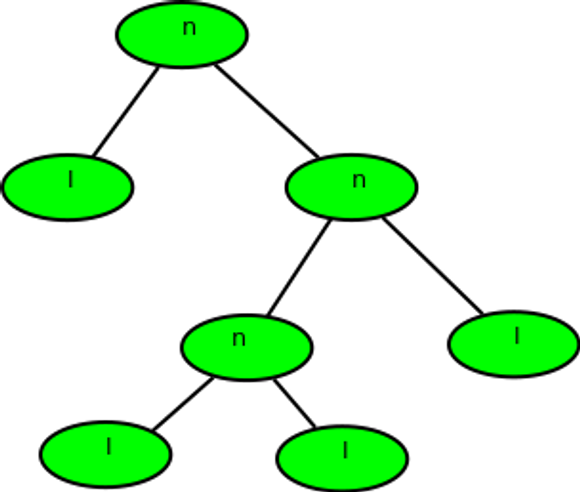
Output : 2

Explanation :



Input : nlnnlll

Output : 3



**Ans:** def calculate\_depth(preorder):

stack = []

depth = 0

for char in preorder:

if char == 'n':

stack.append(char)

elif char == 'l':

depth = max(depth, len(stack))

if stack and stack[-1] == 'n':

stack.pop()

return depth

preorder1 = 'nlnll'

print(calculate\_depth(preorder1))

preorder2 = 'nlnnlll'

print(calculate\_depth(preorder2))

Question-2:

Given a Binary tree, the task is to print the **left view** of the Binary Tree. The left view of a Binary Tree is a set of leftmost nodes for every level.

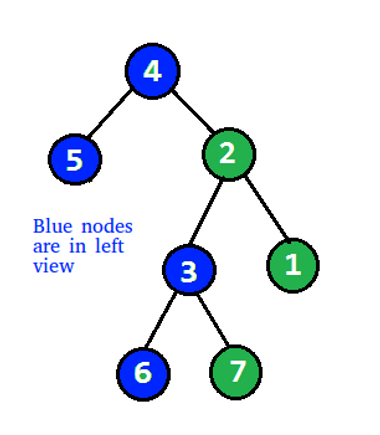
**Examples:**

***Input:***

4  
  
 / \\  
  
 5 2  
  
 / \\  
  
 3 1  
  
 / \\  
  
 6 7

***Output:*** *4 5 3 6*

**Explanation:**



***Input:***

1  
  
 / \\  
  
 2 3  
  
 \\  
  
 4  
  
 \\  
  
 5  
  
 \\  
  
 6

**Output:** 1 2 4 5 6

**Ans:** from collections import deque

class Node:

def \_\_init\_\_(self, data):

self.data = data

self.left = None

self.right = None

def left\_view(root):

if root is None:

return

queue = deque()

queue.append(root)

result = []

while queue:

level\_size = len(queue)

for i in range(level\_size):

node = queue.popleft()

if i == 0:

result.append(node.data)

if node.left:

queue.append(node.left)

if node.right:

queue.append(node.right)

return result

root1 = Node(4)

root1.left = Node(5)

root1.right = Node(2)

root1.right.left = Node(3)

root1.right.right = Node(1)

root1.right.left.left = Node(6)

root1.right.left.right = Node(7)

print(left\_view(root1))

root2 = Node(1)

root2.left = Node(2)

root2.right = Node(3)

root2.left.right = Node(4)

root2.left.right.right = Node(5)

root2.left.right.right.right = Node(6)

print(left\_view(root2))

Question-3:

Given a Binary Tree, print the Right view of it.

The right view of a Binary Tree is a set of nodes visible when the tree is visited from the Right side.

**Examples:**

**Input:**

1  
  
 / \\

2 3

/ \ / \

4 5 6 7

\\  
  
 8

**Output**:

Right view of the tree is 1 3 7 8

**Input:**

1  
  
 /  
  
8

/

7

**Output**:

Right view of the tree is 1 8 7

**Ans:** from collections import deque

class Node:

def \_\_init\_\_(self, data):

self.data = data

self.left = None

self.right = None

def right\_view(root):

if root is None:

return

queue = deque()

queue.append(root)

result = []

while queue:

level\_size = len(queue)

for i in range(level\_size):

node = queue.popleft()

if i == level\_size - 1:

result.append(node.data)

if node.left:

queue.append(node.left)

if node.right:

queue.append(node.right)

return result

root1 = Node(1)

root1.left = Node(2)

root1.right = Node(3)

root1.left.left = Node(4)

root1.left.right = Node(5)

root1.right.left = Node(6)

root1.right.right = Node(7)

root1.right.left.right = Node(8)

print(right\_view(root1))

root2 = Node(1)

root2.left = Node(8)

root2.left.left = Node(7)

print(right\_view(root2))

Question-4:

Given a Binary Tree, The task is to print the **bottom view** from left to right. A node **x** is there in output if x is the bottommost node at its horizontal distance. The horizontal distance of the left child of a node x is equal to a horizontal distance of x minus 1, and that of a right child is the horizontal distance of x plus 1.

**Examples:**

**Input:**

20  
  
 / \\  
  
 8 22  
  
/ \\ \\

5 3 25

/ \\

10 14

**Output:** 5, 10, 3, 14, 25.

**Input:**

20  
  
 / \\  
  
 8 22  
  
/ \\ / \\

5 3 4 25

/ \\  
  
 10 14

**Output:**

5 10 4 14 25.

**Explanation:**

If there are multiple bottom-most nodes for a horizontal distance from the root, then print the later one in the level traversal.

**3 and 4** are both the bottom-most nodes at a horizontal distance of 0, we need to print 4.

**Ans:** from collections import deque

class Node:

def \_\_init\_\_(self, data):

self.data = data

self.left = None

self.right = None

def bottom\_view(root):

if root is None:

return []

queue = deque()

queue.append((root, 0))

hd\_map = {}

while queue:

node, hd = queue.popleft()

hd\_map[hd] = node.data

if node.left:

queue.append((node.left, hd - 1))

if node.right:

queue.append((node.right, hd + 1))

bottom\_view = [hd\_map[hd] for hd in sorted(hd\_map)]

return bottom\_view

root = Node(20)

root.left = Node(8)

root.right = Node(22)

root.left.left = Node(5)

root.left.right = Node(3)

root.right.right = Node(25)

root.left.right.left = Node(10)

root.left.right.right = Node(14)

print(bottom\_view(root))