Q1

def findDepthRec(tree, n, index) :

if (index[0] >= n or tree[index[0]] == 'l'):

return 0

index[0] += 1

left = findDepthRec(tree, n, index)

index[0] += 1

right = findDepthRec(tree, n, index)

return (max(left, right) + 1)

def findDepth(tree, n) :

index = [0]

return findDepthRec(tree, n, index)

if \_\_name\_\_ == '\_\_main\_\_':

tree= "nlnnlll"

n = len(tree)

print(findDepth(tree, n))

Q2

def LeftView(root):

# code here

a=[]

left(root,0,a)

return a

def left(root,level,a):

if not root:

return None

if level==len(a):

a.append(root.data)

left(root.left,level+1,a)

left(root.right,level+1,a)

Q3

class Solution:

def rightView(self,root):

a = []

self.rightside(root,0,a)

return a

def rightside(self,root,cl,a):

if root is None:

return

if len(a) == cl:

a.append(root.data)

self.rightside(root.right,cl+1,a)

self.rightside(root.left,cl+1,a)

return

Q4

class Solution:

def bottomView(self, root):

if root is None:

return

res = []

check = dict()

root.pos = 0

res.append(root)

while len(res) > 0:

root = res[0]

pos = root.pos

check[pos] = root.data

if root.left is not None:

root.left.pos = pos - 1

res.append(root.left)

if root.right is not None:

root.right.pos = pos + 1

res.append(root.right)

res.pop(0)

ans = []

new = check.items()

new = sorted(new)

for i in new:

ans.append(i[1])

return ans