Q1

class Node:

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

self.right = None

self.data = val

self.left = None

prev = None

def BinaryTree2DoubleLinkedList(root):

if root is None:

return root

head = BinaryTree2DoubleLinkedList(root.left);

global prev

if prev is None :

head = root

else:

root.left = prev

prev.right = root

prev = root;

BinaryTree2DoubleLinkedList(root.right);

return head

def print\_dll(head):

while head is not None:

print(head.data, end=" ")

head = head.right

head = BinaryTree2DoubleLinkedList(root)

print\_dll(head)

Q3

class Node:

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

self.data = data

self.left = None

self.right = None

def newNode(data):

temp = Node(data)

return temp

def isleafnode(root):

return not root.left and not root.right

def printRootToLeaf(root):

if not root:

return

path = ""

s = []

s.append((root, path))

while len(s) > 0:

it = s.pop()

root = it[0]

path = it[1]

curr = str(root.data) + " "

path += curr

if isleafnode(root):

print(path)

if root.right:

s.append((root.right, path))

if root.left:

s.append((root.left, path))

printRootToLeaf(root)

Q4

class node:

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

self.data = x

self.left = None

self.right = None

preIndex = 0

def search(arr, strt, end, value):

for i in range(strt, end + 1):

if(arr[i] == value):

return i

def buildTree(inn, pre, inStrt, inEnd):

global preIndex

if(inStrt > inEnd):

return None

tNode = node(pre[preIndex])

preIndex += 1

if (inStrt == inEnd):

return tNode

inIndex = search(inn, inStrt,

inEnd, tNode.data)

tNode.left = buildTree(inn, pre, inStrt,

inIndex - 1)

tNode.right = buildTree(inn, pre,

inIndex + 1, inEnd)

return tNode

def checkPostorder(node, postOrder, index):

if (node == None):

return index

index = checkPostorder(node.left,

postOrder,

index)

index = checkPostorder(node.right,

postOrder,

index)

if (node.data == postOrder[index]):

index += 1

else:

return - 1

return index

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

inOrder = [4, 2, 5, 1, 3]

preOrder = [1, 2, 4, 5, 3]

postOrder = [4, 5, 2, 3, 1]

lenn = len(inOrder)

root = buildTree(inOrder, preOrder,

0, lenn - 1)

index = checkPostorder(root, postOrder, 0)

if (index == lenn):

print("Yes")

else:

print("No")