Q1

class newNode:

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

self.key = key

self.left = self.right = None

def findLargestSubtreeSumUtil(root, ans):

if (root == None):

return 0

currSum = (root.key +

findLargestSubtreeSumUtil(root.left, ans) +

findLargestSubtreeSumUtil(root.right, ans))

ans[0] = max(ans[0], currSum)

return currSum

def findLargestSubtreeSum(root):

if (root == None):

return 0

ans = [-999999999999]

findLargestSubtreeSumUtil(root, ans)

return ans[0]

root = newNode(1)

root.left = newNode(-2)

root.right = newNode(3)

root.left.left = newNode(4)

root.left.right = newNode(5)

root.right.left = newNode(-6)

root.right.right = newNode(2)

print(findLargestSubtreeSum(root))

Q2

def LevelOrder(root, data):

if(root == None):

root = getNode(data)

return root

if(data <= root.data):

root.left = LevelOrder(root.left, data)

else:

root.right = LevelOrder(root.right, data)

return root

def constructBst(arr, n):

if(n == 0):

return None

root = None

for i in range(0, n):

root = LevelOrder(root, arr[i])

return root

def inorderTraversal(root):

if (root == None):

return None

inorderTraversal(root.left)

print(root.data, end=" ")

inorderTraversal(root.right)

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

arr = [7, 4, 12, 3, 6, 8, 1, 5, 10]

n = len(arr)

root = constructBst(arr, n)

print("Inorder Traversal: ", end="")

root = inorderTraversal(root)

Q3

class NodeDetails:

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

self.data = data

self.min = min

self.max = max

def levelOrderIsOfBST(arr, n):

if n == 0:

return True

q = []

i = 0

newNode = NodeDetails(arr[i], INT\_MIN, INT\_MAX)

i += 1

q.append(newNode)

while i != n and len(q) != 0:

temp = q.pop(0)

if i < n and (arr[i] < temp.data and

arr[i] > temp.min):

newNode = NodeDetails(arr[i], temp.min, temp.data)

i += 1

q.append(newNode)

if i < n and (arr[i] > temp.data and

arr[i] < temp.max):

newNode = NodeDetails(arr[i], temp.data, temp.max)

i += 1

q.append(newNode)

if i == n:

return True

return False

if \_\_name\_\_ == "\_\_main\_\_":

arr = [7, 4, 12, 3, 6, 8, 1, 5, 10]

n = len(arr)

if levelOrderIsOfBST(arr, n):

print("Yes")

else:

print("No")