```
'''

    Time complexity: O(N)
    Space complexity: O(H)

    where N is the number of nodes in the input tree
    and H is the height of the input tree

'''

from sys import stdin, setrecursionlimit
import queue

setrecursionlimit(10 ** 6)


MIN_VALUE = -9999999999
MAX_VALUE = 9999999999


#Following is the structure used to represent the Binary Tree Node
class BinaryTreeNode:
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None


class Pair :

        def __init__(self, minimum, maximum) :
                self.minimum = minimum
                self.maximum = maximum



def getMinAndMax(root) :

        if root is None :
                return Pair(MAX_VALUE, MIN_VALUE)


        leftPair = getMinAndMax(root.left)
        rightPair = getMinAndMax(root.right)

        minimum = min(root.data, leftPair.minimum, rightPair.minimum)
        maximum = max(root.data, leftPair.maximum, rightPair.maximum)

        return Pair(minimum, maximum)




#Taking level-order input using fast I/O method
def takeInput():
    levelOrder = list(map(int, stdin.readline().strip().split(" ")))
    start = 0

    length = len(levelOrder)

    if length == 1 :
        return None

    root = BinaryTreeNode(levelOrder[start])
    start += 1

    q = queue.Queue()
    q.put(root)

    while not q.empty():
        currentNode = q.get()

        leftChild = levelOrder[start]
        start += 1

        if leftChild != -1:
            leftNode = BinaryTreeNode(leftChild)
            currentNode.left =leftNode
            q.put(leftNode)

        rightChild = levelOrder[start]
        start += 1

        if rightChild != -1:
            rightNode = BinaryTreeNode(rightChild)
            currentNode.right =rightNode
            q.put(rightNode)

    return root


def printLevelWise(root):
    if root is None:
        return

    inputQ = queue.Queue()
    outputQ = queue.Queue()
    inputQ.put(root)

    while not inputQ.empty():

        while not inputQ.empty():

            curr = inputQ.get()
            print(curr.data, end=' ')
            if curr.left!=None:
                outputQ.put(curr.left)
            if curr.right!=None:
                outputQ.put(curr.right)

        print()
        inputQ, outputQ = outputQ, inputQ


# Main
root = takeInput()

pair = getMinAndMax(root)
print(str(str(pair.minimum) + " " + str(pair.maximum)))
```