Question :

Arun is the class teacher of class 10. You are having the heights of all the boys and girls but separately. Help Arun to find the median height of the class.

Tags :

LinkedList, Array

Input Description :

First line consists of number of boys
Second line consists of height of boys
Third line consists of number of girls
Fourth line consists of height of girls

Output Description :

Median Height of the array

Solution:

```
# Class Teacher - Median Height

class Node:
    '''Initializes a Node for Singly Linked List'''
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:
    ''' Initializes a Linked List'''
    def __init__(self):
        self.head = None

    def printLinkedList(self):
        ''' Prints LinkedList '''
        temp = self.head
        while temp is not None:
            if temp.next is not None:
                print(temp.data, end = " ")
```

```
            else:
                print(temp.data, end = "")
            temp = temp.next


def createLinkedList(lst, n):
    ''' Creates a LinkedList '''
    ll = LinkedList()
    temp = ll.head
    for i in range(n):
        new_node = Node(lst[i])
        if ll.head is None:
            ll.head = new_node
            temp = new_node
        else:
            temp.next = new_node
            temp = new_node

    return ll

def merge_linked_list(ll1, ll2):
    ll3 = LinkedList()
    temp1 = ll1.head
    temp2 = ll2.head
    while True:
        if temp1.data <= temp2.data:
            new_node = Node(temp1.data)
            temp1 = temp1.next
        else:
            new_node = Node(temp2.data)
            temp2 = temp2.next
        if ll3.head is None:
            ll3.head = new_node
            temp = new_node
        else:
            temp.next = new_node
```

```
            temp = new_node
        if temp1 is None or temp2 is None:
            break
    while temp1 is not None:
        new_node = Node(temp1.data)
        temp1 = temp1.next
        if ll3.head is None:
            ll3.head = new_node
            temp = new_node
        else:
            temp.next = new_node
            temp = new_node

    while temp2 is not None:
        new_node = Node(temp2.data)
        temp2 = temp2.next
        if ll3.head is None:
            ll3.head = new_node
            temp = new_node
        else:
            temp.next = new_node
            temp = new_node

    return ll3

# Median of Linked List
def median(ll, n):
    if n&1:
        mid = [(n+1) // 2]
    else:
        mid = [n // 2, (n//2) + 1]
    count = 1
    temp = ll.head
    median = []
    while temp is not None and len(mid) != 0:
        if count in mid:
```

```
            median.append(temp.data)
            mid.pop(0)
        count += 1
        temp = temp.next

    if len(median) == 2:
        return sum(median) / 2
    else:
        return median[0]


n = int(input())
ll1 = createLinkedList([int(x) for x in input().split()], n)

m = int(input())
ll2 = createLinkedList([int(x) for x in input().split()], m)

merged_linkedlist = merge_linked_list(ll1, ll2)

print(median(merged_linkedlist, n+m))
```

Test Cases :

Test Case 1:
Input
5
1 2 3 4 5
5
3 4 5 6 7
Output
4.0

Test Case 2:
Input
3
1 5 9

7
2 3 4 6 7 8 10

Test Case 3:
Input
7
11 12 23 34 35 41 45
4
10 20 30 40
Output
30

Test Case 4:
Input
2
1 2
2
1 2
Output
1.5

Test Case 5:
Input
10
1 3 5 7 9 11 13 15 17 19
9
2 6 8 16 23 42 54 67 89
Output
13