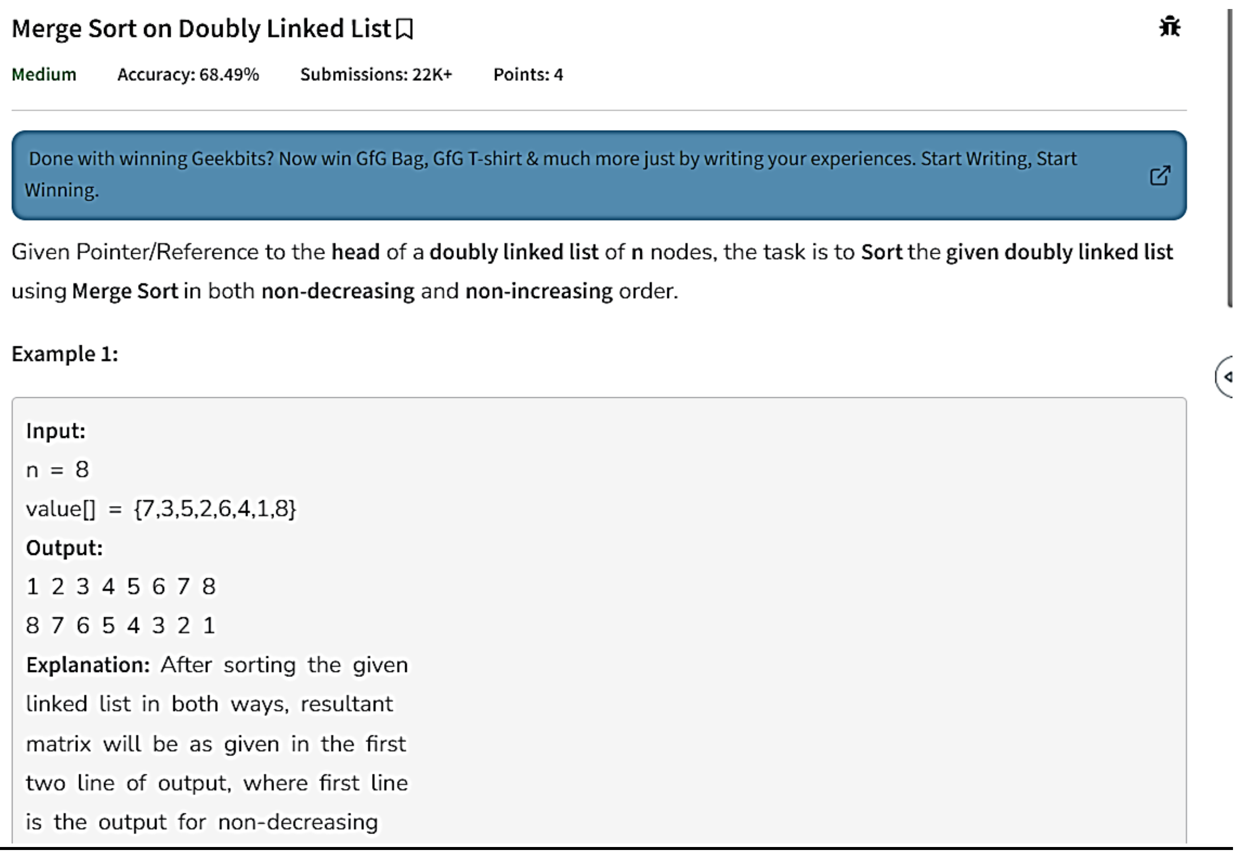
PROBLEM



Merge Sort on Doubly Linked List

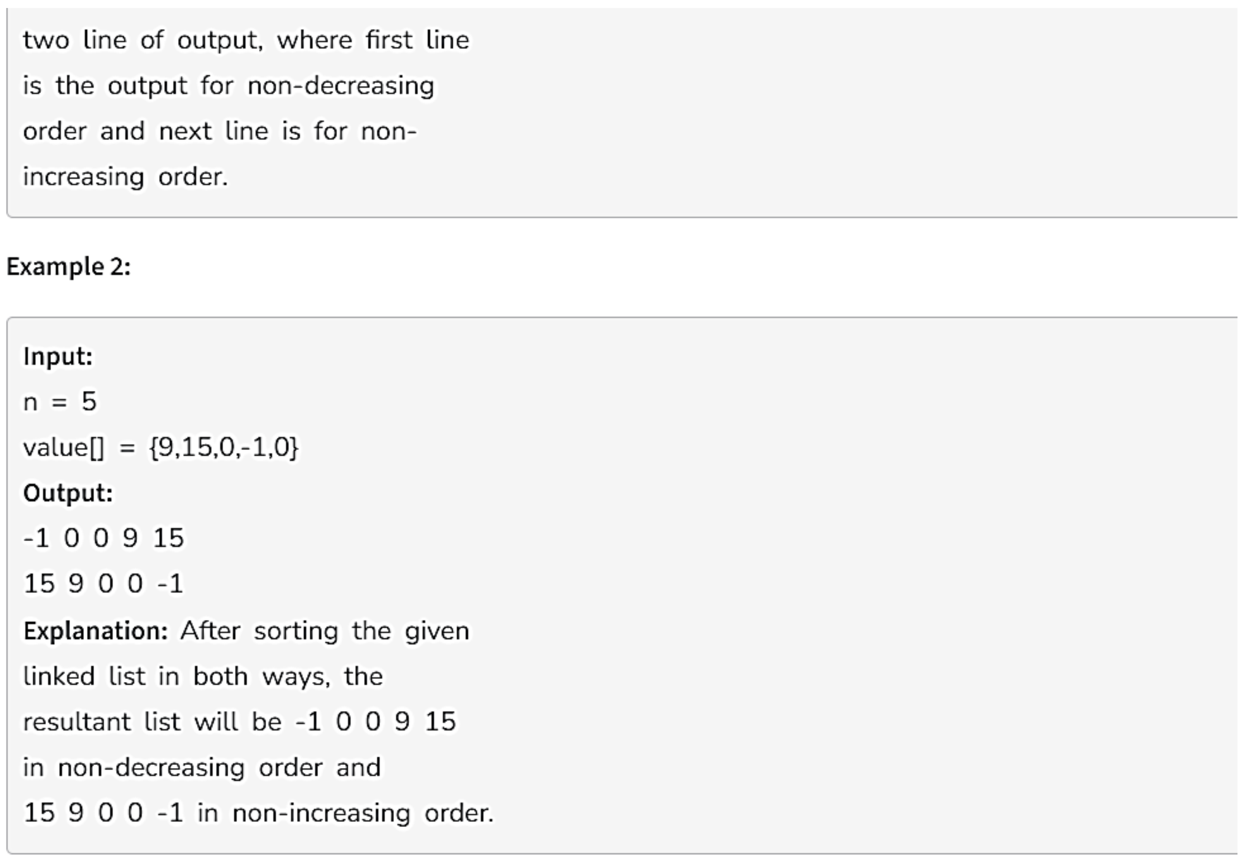
Medium   Accuracy: 68.49%   Submissions: 22K+   Points: 4

Done with winning Geekbits? Now win GfG Bag, GfG T-shirt & much more just by writing your experiences. Start Writing, Start Winning.

Given Pointer/Reference to the **head** of a **doubly linked list** of **n** nodes, the task is to **Sort** the **given doubly linked list** using **Merge Sort** in both **non-decreasing** and **non-increasing** order.

Example 1:

```
Input:
n = 8
value[] = {7,3,5,2,6,4,1,8}
Output:
1 2 3 4 5 6 7 8
8 7 6 5 4 3 2 1
Explanation: After sorting the given
linked list in both ways, resultant
matrix will be as given in the first
two line of output, where first line
is the output for non-decreasing
```



two line of output, where first line
is the output for non-decreasing

```
order and next line is for non-
increasing order.
```

Example 2:

```
Input:
n = 5
value[] = {9,15,0,-1,0}
Output:
-1 0 0 9 15
15 9 0 0 -1
Explanation: After sorting the given
linked list in both ways, the
resultant list will be -1 0 0 9 15
in non-decreasing order and
15 9 0 0 -1 in non-increasing order.
```

Your Task:

The task is to complete the function **sortDoubly()** which takes reference to the **head** of the doubly linked and **Sort** the given doubly linked list using **Merge Sort** in both **non-decreasing** and **non-increasing**. The **printing** is done **automatically** by the **driver code**.

Expected Time Complexity: O(nlogn)
Expected Space Complexity: O(logn)

Constraints:
$1 <= n <= 10^4$
$0 <= values[i] <= 10^5$

CODE

```
#User function Template for python3

'''
class Node:
        def __init__(self, data):
                self.data = data
                self.next = None
                self.prev = None
'''

class Solution():
#Function to sort the given doubly linked list using Merge Sort.
  def sortDoubly(self,head:'Node') -> 'Node':
    def merge(l1: Node, l2: Node) -> Node:
      dummy = Node(-1)
      node = dummy
      while l1 and l2:
        if l1.data <= l2.data:
          node.next, node, l1 = l1, l1, l1.next
        else:
          node.next, node, l2 = l2, l2, l2.next
      # Attaching any remaining nodes
      node.next = l1 or l2
      return dummy.next

    def mergesort(l1: Node, n: int) -> Node:
      if n == 1:
        return l1
      n_half = n // 2
      l1_tail = l1
      for _ in range(n_half - 1):
        l1_tail = l1_tail.next
      l2, l1_tail.next = l1_tail.next, None
      l1 = mergesort(l1, n_half)
      l2 = mergesort(l2, n - n_half)
      return merge(l1, l2)

    n, node = 0, head
    while node:
      n += 1
      node = node.next
    l = mergesort(head, n)
    # Fixing prevs
    prev, node = None, l
    while node:
      node.prev, prev, node = prev, node, node.next
    return l
  #Your code her
```