TASK REPORT

ALGORITHMS AND DATA STRUCTURE

Linked List Problems

Created by:

Onic Agustino

L200234275

X

Date : March 29, 2025

INFORMATICS ENGINEERING

FACULTY OF COMMUNICATION AND INFORMATICS

UNIVERSITAS MUHAMMADIYAH SURAKARTA

2024/2025

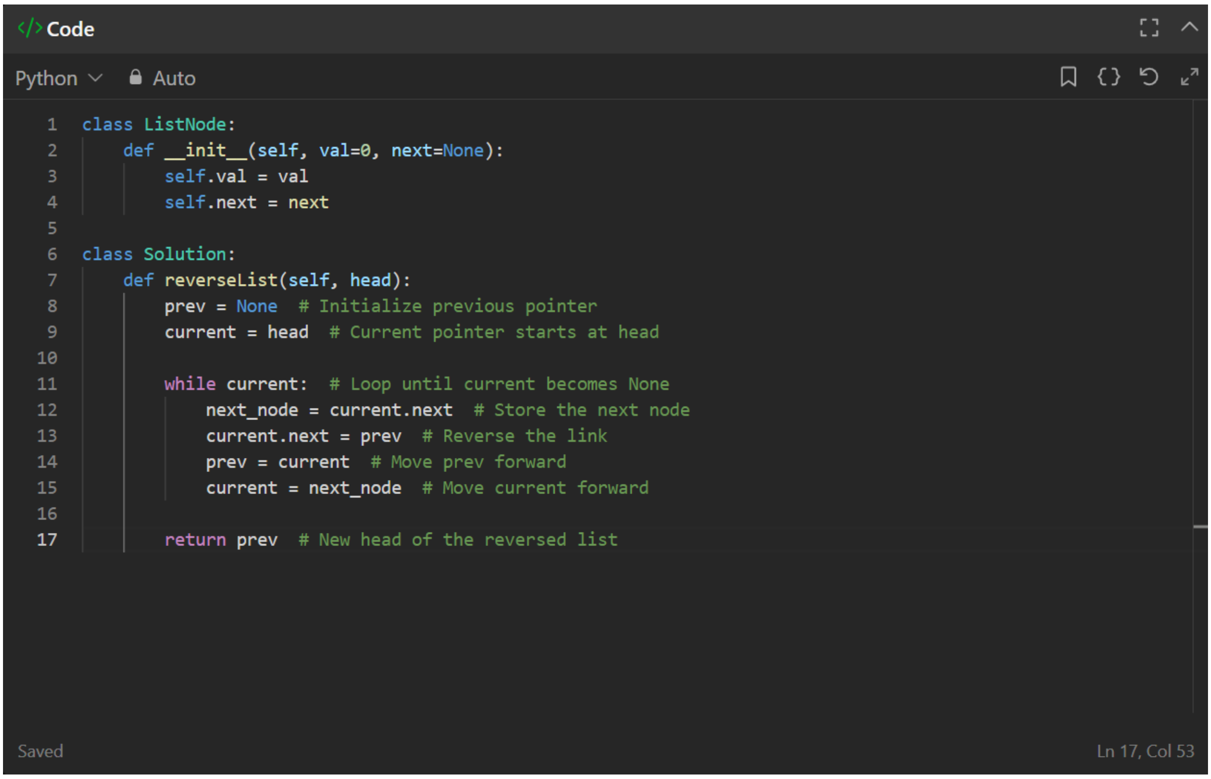
Leetcode

1. Reserve Linked List

- **Problem Explanation**

  Given the head of a singly linked list, reverse the list and return its new head. The input is a linked list, and the output is linked list with the order of elements reversed.

- **Solution Approach**
  - We use iterative approach with two pointers : prev (stores the previous node) and curr (stores the current node).
  - We traverse through the linked list while updating pointers to reverse the direction of each node.
  - At the end of the traversal, prev will be the new head of the reversed list.
- **Code Implementation**



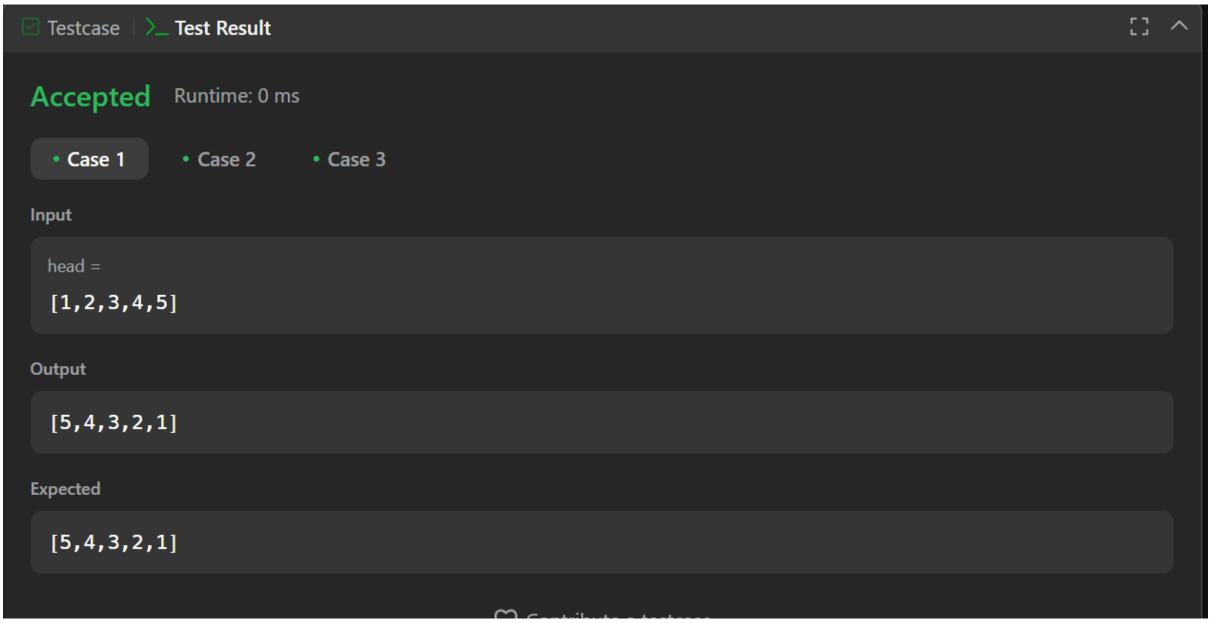
```python
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

class Solution:
    def reverseList(self, head):
        prev = None  # Initialize previous pointer
        current = head  # Current pointer starts at head

        while current:  # Loop until current becomes None
            next_node = current.next  # Store the next node
            current.next = prev  # Reverse the link
            prev = current  # Move prev forward
            current = next_node  # Move current forward

        return prev  # New head of the reversed list
```

- **Output**



Accepted Runtime: 0 ms

Input

head =

```
[1,2,3,4,5]
```

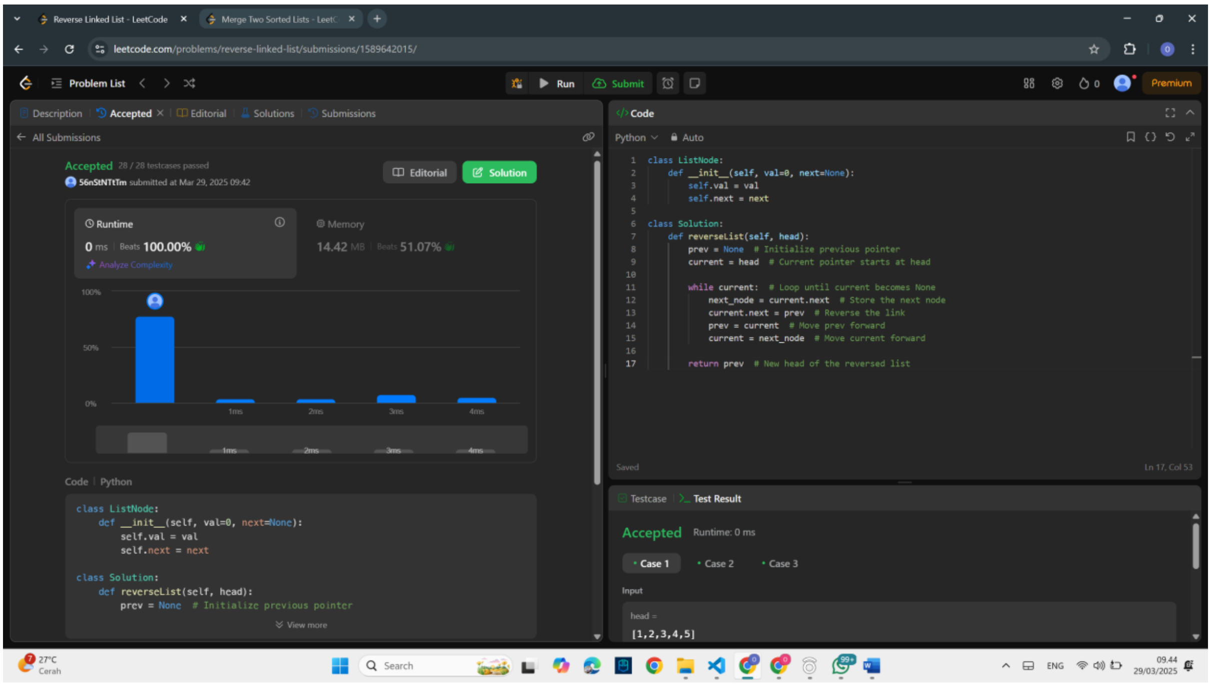
Output

```
[5,4,3,2,1]
```

Expected

```
[5,4,3,2,1]
```

- **Screenshot Submission**



Accepted 28 / 28 testcases passed

56nStNTtTm submitted at Mar 29, 2025 09:42

Runtime 0 ms | Beats 100.00%

Memory 14.42 MB | Beats 51.07%

Code | Python

```python
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

class Solution:
    def reverseList(self, head):
        prev = None  # Initialize previous pointer
        current = head  # Current pointer starts at head

        while current:  # Loop until current becomes None
            next_node = current.next  # Store the next node
            current.next = prev  # Reverse the link
            prev = current  # Move prev forward
            current = next_node  # Move current forward

        return prev  # New head of the reversed list
```

Accepted Runtime: 0 ms

Input

head =

```
[1,2,3,4,5]
```

\

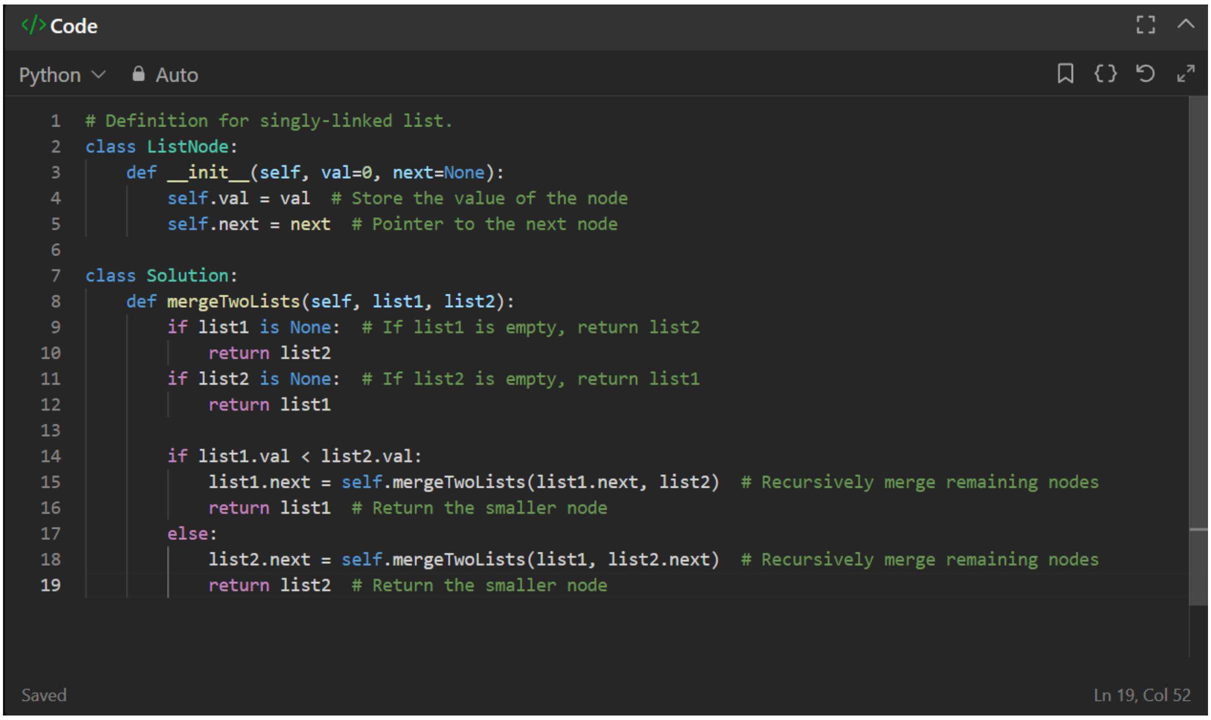
2. Merge Two Sortes Lists

- **Problem Explanation**

  Given the heads of two sorted linked list, merge them into one sorted linked list.

- **Solution Approach**
  - We compare the values of nodes from both lists and attach the smaller one to the merged list
  - This is done recursively by choosing the smaller node and calling the function for the remaining nodes.

- **Code Implementation**

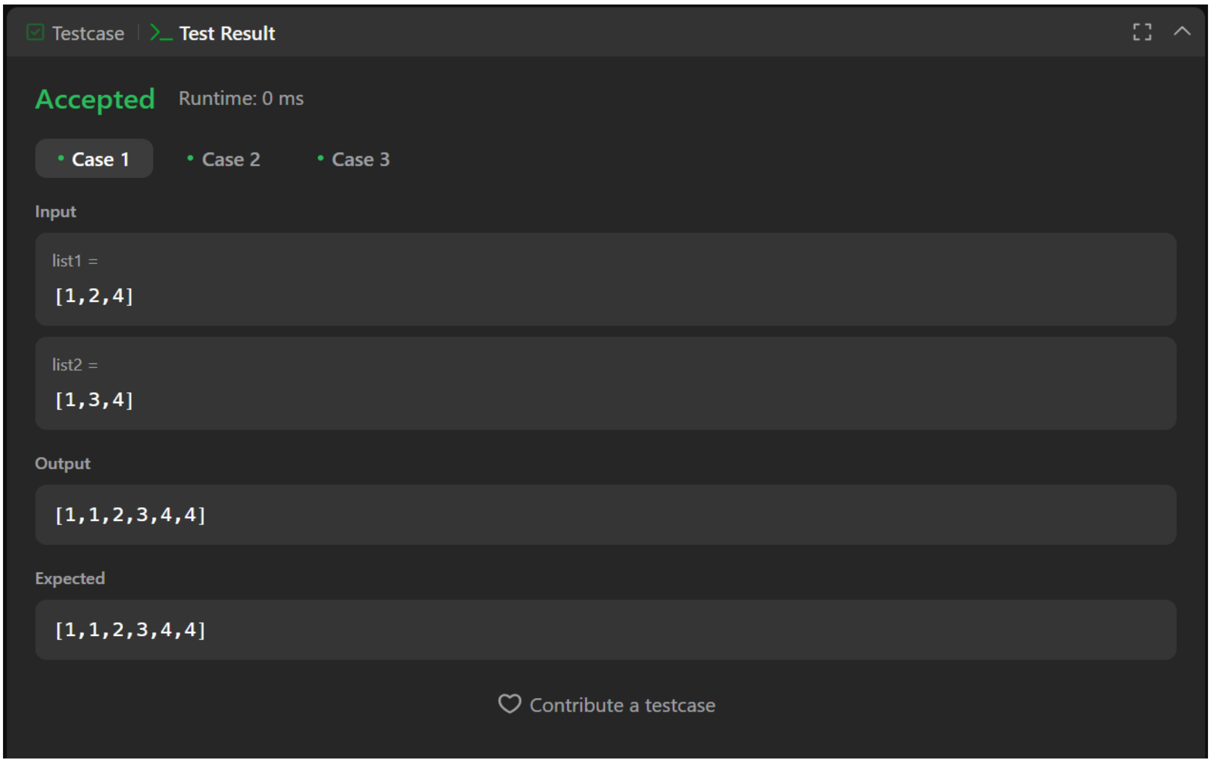


```python
# Definition for singly-linked list.
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val  # Store the value of the node
        self.next = next  # Pointer to the next node

class Solution:
    def mergeTwoLists(self, list1, list2):
        if list1 is None:  # If list1 is empty, return list2
            return list2
        if list2 is None:  # If list2 is empty, return list1
            return list1

        if list1.val < list2.val:
            list1.next = self.mergeTwoLists(list1.next, list2)  # Recursively merge remaining nodes
            return list1  # Return the smaller node
        else:
            list2.next = self.mergeTwoLists(list1, list2.next)  # Recursively merge remaining nodes
            return list2  # Return the smaller node
```

- **Output**



Accepted Runtime: 0 ms

Input

list1 =

```
[1,2,4]
```

list2 =

```
[1,3,4]
```

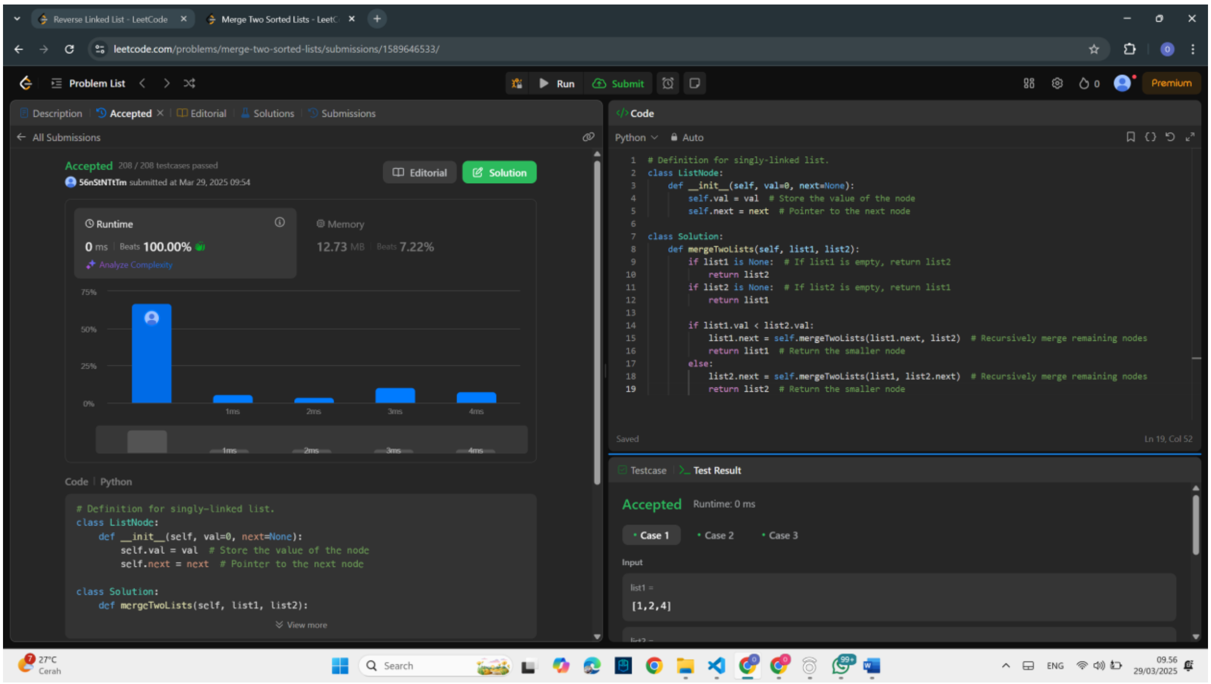
Output

```
[1,1,2,3,4,4]
```

Expected

```
[1,1,2,3,4,4]
```

- **Screenshot Submission**



Accepted 208 / 208 testcases passed

56nStNTtTm submitted at Mar 29, 2025 09:54

Runtime: 0 ms | Beats 100.00%

Memory: 12.73 MB | Beats 7.22%

```python
# Definition for singly-linked list.
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val  # Store the value of the node
        self.next = next  # Pointer to the next node

class Solution:
    def mergeTwoLists(self, list1, list2):
        if list1 is None:  # If list1 is empty, return list2
            return list2
        if list2 is None:  # If list2 is empty, return list1
            return list1

        if list1.val < list2.val:
            list1.next = self.mergeTwoLists(list1.next, list2)  # Recursively merge remaining nodes
            return list1  # Return the smaller node
        else:
            list2.next = self.mergeTwoLists(list1, list2.next)  # Recursively merge remaining nodes
            return list2  # Return the smaller node
```

Accepted Runtime: 0 ms

Input

list1 =

```
[1,2,4]
```