



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Experiment- 2A

Student Name: Deepanshu

UID: 22BCS16443

Branch: BE-CSE

Section/Group: NTPP 602-A

Semester: 6TH

Date of Performance: 10/02/25

Subject Name: AP Lab-2

Subject Code: 22CSH-352

1. TITLE:

Remove Duplicates from Sorted List

2. AIM:

Given the head of a sorted linked list, *delete all duplicates such that each element appears only once*. Return *the linked list sorted as well*.

3. Algorithm

- Start with the head of the linked list.
- Iterate through the linked list while the next node is not None.
- If the current node's value is equal to the next node's value, update the `next` pointer to skip the duplicate node.
- Otherwise, move to the next node.

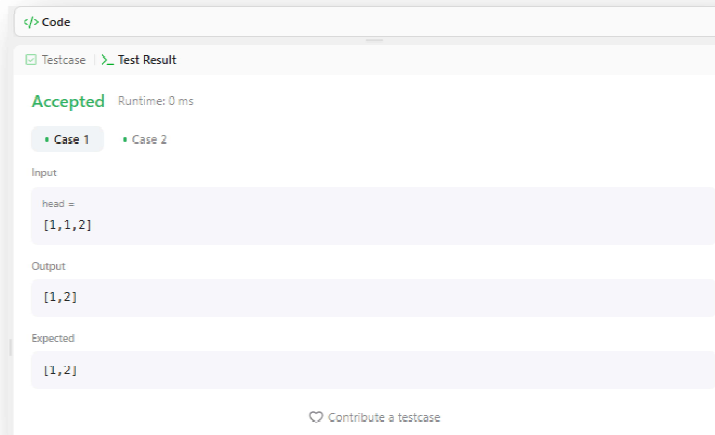
Implementation/Code

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

```
class Solution:
    def deleteDuplicates(self, head):
        current = head
        while current and current.next:
            if current.val == current.next.val:
                current.next = current.next.next
            else:
                current = current.next
        return head
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Output:



Time Complexity : $O(n)$

Space Complexity : $O(1)$

Learning Outcomes:-

- Learn how to iterate through a linked list efficiently.
- Understand how the sorted order helps in detecting duplicates efficiently.



Experiment - 2B

Student Name: Deepanshu

UID: 22BCS16443

Branch: BE-CSE

Section/Group: NTPP- 602(A)

Semester: 6TH

Date of Performance: 10/02/25

Subject Name: AP Lab-2

Subject Code: 22CSH-352

1. TITLE:

Reverse Linked List.

2. AIM:

Given the head of a singly linked list, reverse the list, and return *the reversed list*.

3. Algorithm

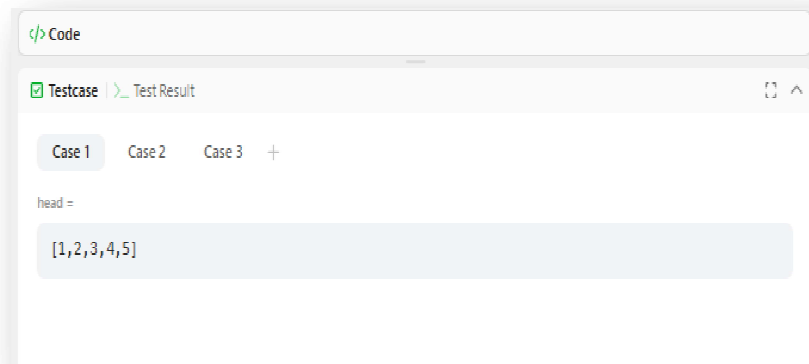
- Set prev = None.
- Set current = head.
- Store the next node (next_node = current.next) before modifying links.
- Reverse the link (current.next = prev) to point backward.
- Move prev and current one step forward.
- After the loop, prev will be the new head.

Implementation/Code:

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
class Solution:
    def reverseList(self, head):
        prev = None
        current = head
        # Traverse the linked list
        while current:
            next_node = current.next
            current.next = prev
            prev = current
            current = next_node
        return prev
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Output:



The screenshot shows a code editor window with a tab labeled 'Code'. Below the tab, there is a 'Testcase' section with a green checkmark icon and a 'Test Result' section with a green arrow icon. Under the 'Testcase' section, there are three tabs: 'Case 1', 'Case 2', and 'Case 3', with a plus sign to the right. The 'Case 1' tab is selected. Below the tabs, the text 'head =' is displayed, followed by a light blue box containing the array '[1,2,3,4,5]'.

Time Complexity : $O(N)$

Space Complexity : $O(1)$

Learning Outcomes:-

- Learn how to modify `next` pointers to reverse the direction of a linked list.
- Understand how to efficiently reverse a linked list using only a few pointers (`prev`, `current`, `next_node`).



DEPARTMENT OF **COMPUTER SCIENCE & ENGINEERING**

Discover. Learn. Empower.