



Experiment 3

Student Name: Rahul Prasad

UID: 22BET10167

Branch: IT

Section/Group: 701/A

Semester: 6th

Date : 12/02/2025

Subject Name: Advanced Programming Lab - 2

Subject Code: 22ITP-351

1. Problem 1:

➤ Remove Duplicates from Sorted List:

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.

➤ Code:

```
class Solution {  
  
    public ListNode deleteDuplicates(ListNode head) {  
  
        if (head == null) {  
  
            return null;  
  
        }  
  
        ListNode current = head;  
  
        while (current != null && current.next != null) {  
  
            if (current.val == current.next.val) {  
  
                current.next = current.next.next;  
  
            }  
  
        }  
  
    }  
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        } else {  
            current = current.next;  
        }  
    }  
  
    return head;  
}  
}
```

➤ Output

☒ Testcase | [Test Result](#)

Accepted Runtime: 0 ms

- Case 1
- Case 2

Input
head =
[1,1,2,3,3]

Output
[1,2,3]

Expected
[1,2,3]

2. Problem 2:

➤ Reverse Linked List:

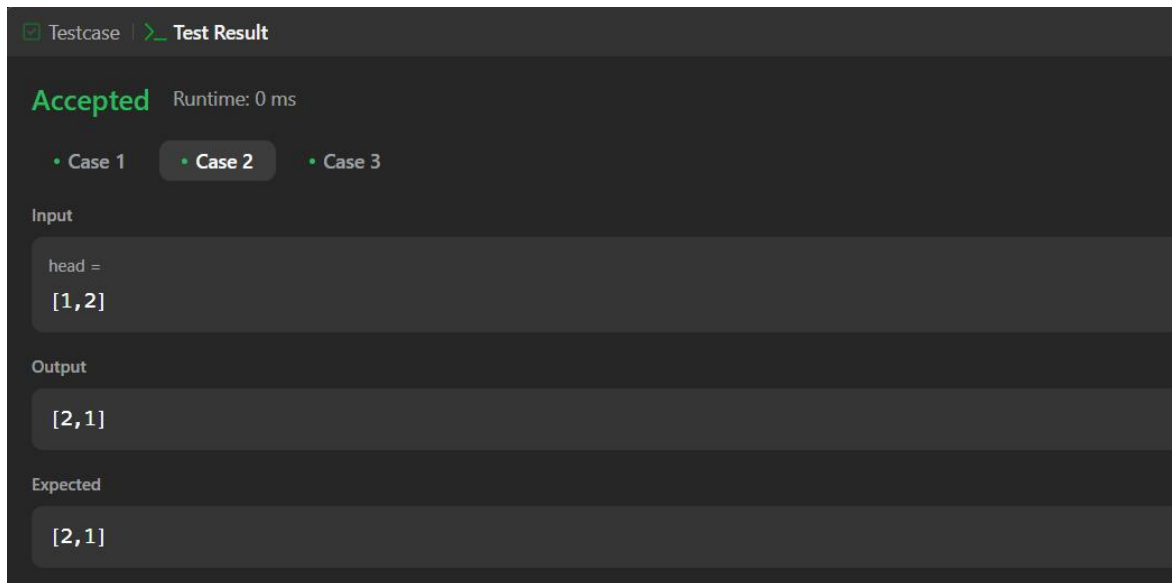
Given the head of a singly linked list, reverse the list, and return the reversed list.

➤ Code:

```
class Solution {  
  
    public ListNode reverseList(ListNode head) {  
  
        ListNode temp = head;  
  
        ListNode prev = null;  
  
        while (temp != null) {  
  
            ListNode front = temp.next;  
  
            temp.next = prev;  
  
            prev = temp;  
  
            temp = front;  
  
        }  
  
        return prev;  
  
    }  
}
```



➤ Output:



3. Problem - 3:

➤ Delete the Middle Node of a Linked List:

You are given the head of a linked list. Delete the middle node, and return the head of the modified linked list.

The middle node of a linked list of size n is the $\lfloor n / 2 \rfloor$ th node from the start using 0-based indexing, where $\lfloor x \rfloor$ denotes the largest integer less than or equal to x .

For $n = 1, 2, 3, 4$, and 5 , the middle nodes are $0, 1, 1, 2$, and 2 , respectively.

➤ Code:

```
class Solution {
    public ListNode deleteMiddle(ListNode head) {
        if (head == null || head.next == null) {
            return null;
        }
        ListNode fast = head.next.next;
        ListNode slow = head;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        while (fast != null && fast.next != null) {  
            fast = fast.next.next;  
            slow = slow.next;  
        }  
  
        slow.next = slow.next.next;  
        return head;  
    }  
}
```

➤ **Output:**

A screenshot of a web-based testing interface. At the top, there are two tabs: 'Testcase' (with a green checkmark) and 'Test Result' (with a green arrow). Below the tabs, the word 'Accepted' is displayed in green, followed by 'Runtime: 0 ms'. There are three radio buttons for selecting a test case: 'Case 1', 'Case 2' (which is selected), and 'Case 3'. Below this, there are three sections: 'Input' showing 'head = [1,2,3,4]', 'Output' showing '[1,2,4]', and 'Expected' showing '[1,2,4]'. All sections have a dark background with light-colored text.

4. Problem - 4:

➤ Merge Two Sorted Lists:

You are given the heads of two sorted linked lists list1 and list2.

Merge the two lists into one sorted list. The list should be made by splicing together the nodes of the first two lists.

Return the head of the merged linked list.

➤ Code:

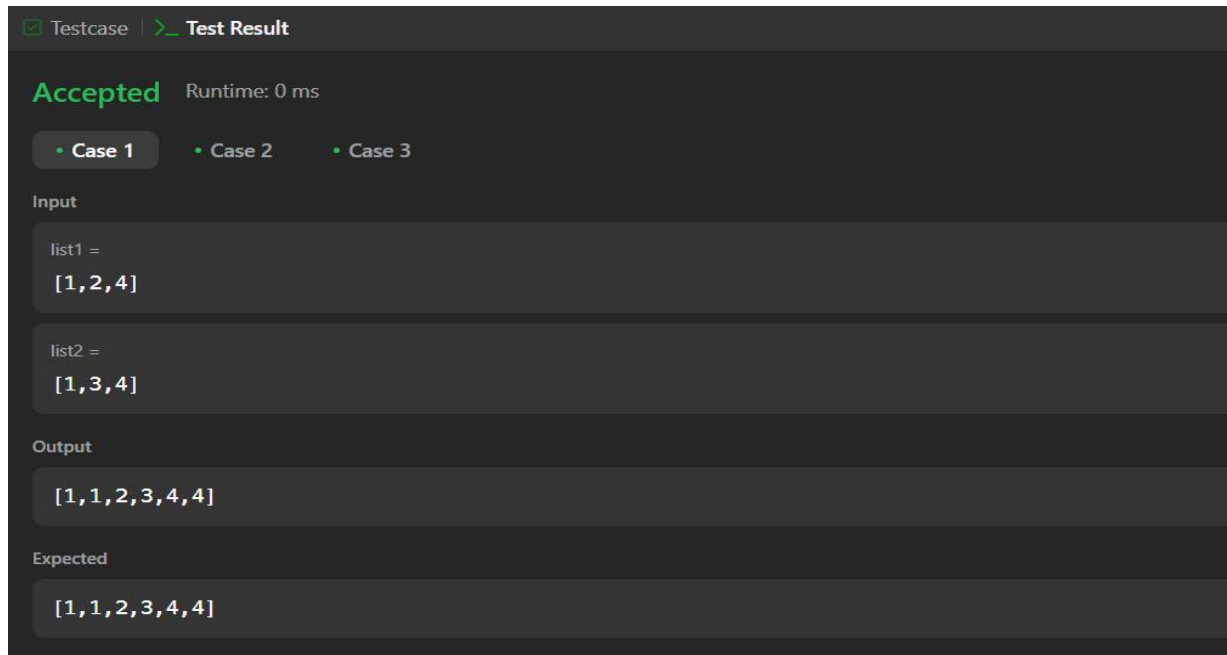
```
class Solution {
    public ListNode mergeTwoLists(ListNode list1, ListNode list2) {
        ListNode list = new ListNode();
        ListNode temp = list;

        while (list1 != null && list2 != null) {
            if (list1.val <= list2.val) {
                temp.next = list1;
                list1 = list1.next;
            } else {
                temp.next = list2;
                list2 = list2.next;
            }
            temp = temp.next;
        }

        if (list1 != null) {
            temp.next = list1;
        } else if (list2 != null) {
            temp.next = list2;
        }

        return list.next;
    }
}
```

➤ Output:



5. Problem - 5:

➤ Remove Duplicates from Sorted List II:

Given the head of a sorted linked list, delete all nodes that have duplicate numbers, leaving only distinct numbers from the original list. Return the linked list sorted as well.

➤ Code:

```
class Solution {
    public ListNode deleteDuplicates(ListNode head) {
        if (head == null || head.next == null) {
            return head;
        }

        ListNode dummy = new ListNode(0, head);
        ListNode slow = dummy;
        ListNode fast = head;

        while (fast != null) {
            if (fast.next != null && fast.val == fast.next.val) {
                while (fast.next != null && fast.val == fast.next.val) {
                    fast = fast.next;
                }
            }
            slow.next = fast;
            slow = fast;
            fast = fast.next;
        }
        return dummy.next;
    }
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        }
        slow.next = fast.next;
    } else {
        slow = fast;
    }
    fast = fast.next;
}

return dummy.next;
}
```

➤ Output:

☒ Testcase | [Test Result](#)

Accepted Runtime: 0 ms

- Case 1
- **Case 2**

Input
head =
[1,1,1,2,3]

Output
[2,3]

Expected
[2,3]

6. Problem - 6:

➤ Linked List Cycle:

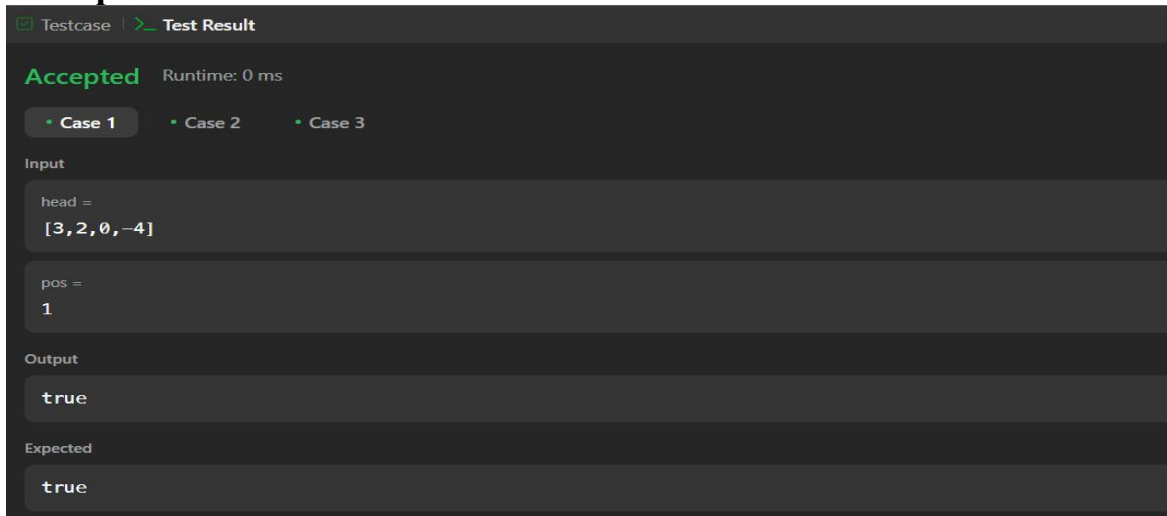
Given head, the head of a linked list, determine if the linked list has a cycle in it. There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the next pointer. Internally, pos is used to denote the index of the node that tail's next pointer is connected to. Note that pos is not passed as a parameter.

Return true if there is a cycle in the linked list. Otherwise, return false.

➤ Code:

```
public class Solution {  
    public boolean hasCycle(ListNode head) {  
        ListNode slow = head;  
        ListNode fast = head;  
        while (fast != null && fast.next != null) {  
            slow = slow.next;  
            fast = fast.next.next;  
            if (slow == fast) {  
                return true;  
            }  
        }  
        return false;  
    }  
}
```

➤ Output:



The screenshot shows a test result interface with a dark theme. At the top, there are tabs for 'Testcase' and 'Test Result', with 'Test Result' being the active tab. Below the tabs, the status 'Accepted' is displayed in green, followed by 'Runtime: 0 ms'. There are three test cases listed: 'Case 1', 'Case 2', and 'Case 3', with 'Case 1' being the selected one. Under the 'Input' section, there are two fields: 'head =' with the value '[3,2,0,-4]' and 'pos =' with the value '1'. Under the 'Output' section, the value 'true' is displayed. At the bottom, under the 'Expected' section, the value 'true' is also displayed.



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

❖ Learning Outcomes:

- Understand and implement various operations on linked lists, including deletion, reversal, and merging.
- Apply two-pointer techniques to solve problems efficiently, such as detecting cycles and removing duplicates.
- Improve problem-solving skills by handling edge cases like empty lists, duplicate values, and middle node deletion.
- Enhance understanding of time and space complexity for different linked list algorithms.
- Gain hands-on experience with real-world coding patterns used in technical interviews and software development.