Experiment 2

Student Name: Shristi Rai	**UID: 22BCS15330**
Branch: BE-CSE	**Section/Group: NTPP-602-A**
Semester: 6th	**Date: 20-01-25**
Subject Name: AP LAB-II	**Subject Code: 22CSP-351**

1. Aim:

Problem 1.2.1: Remove duplicates from Sorted List

- **Problem Statement:** 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.

Problem 1.2.2: Reverse Linked List

- **Problem Statement:** Given the head of a singly linked list, reverse the list, and return the reversed list.

Problem 1.2.3: Top K Frequent Elements

- **Problem Statement**: Given an integer array nums and an integer k, return *the* k *most frequent elements*. You may return the answer in **any order**.

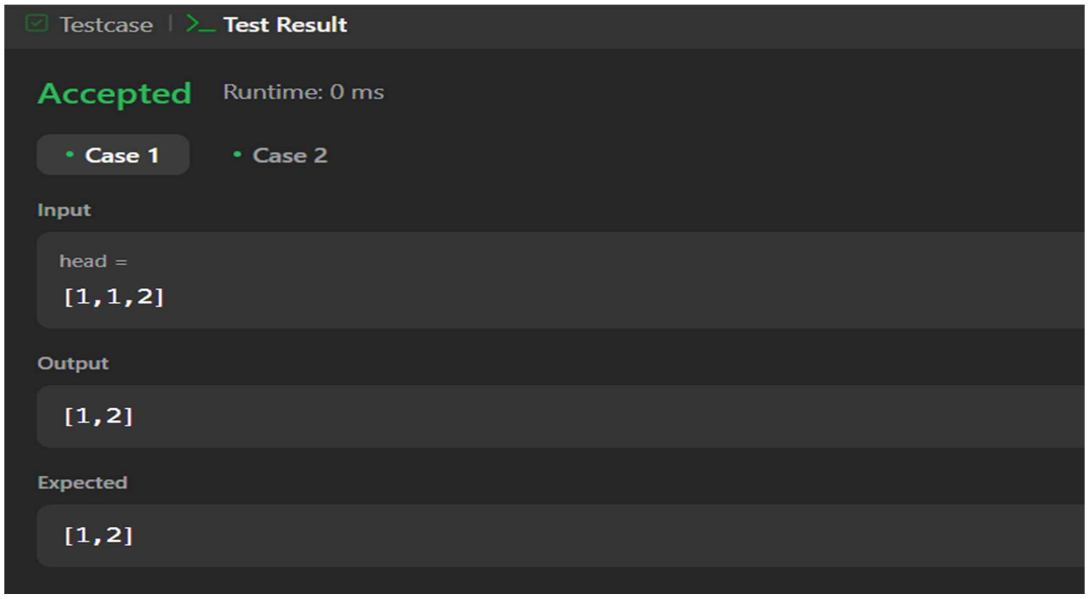
2. Objective:

- Enhance problem-solving skills for handling common algorithmic challenges.
- Count the frequency of each element using a hash map.
- Use an array of lists (bucket sort) to store numbers by their frequency.
- Extract the top k frequent elements from the highest frequency buckets.

Code: 1.2.1

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

        ListNode current = head;
        while (current != null && current.next != null) {
            // If the current node's value equals the next node's value
            if (current.val == current.next.val) {
                // Skip the next node (remove the duplicate)
                current.next = current.next.next;
            } else {
                current = current.next;
            }
        }
        return head;
    }
}
```

Output:



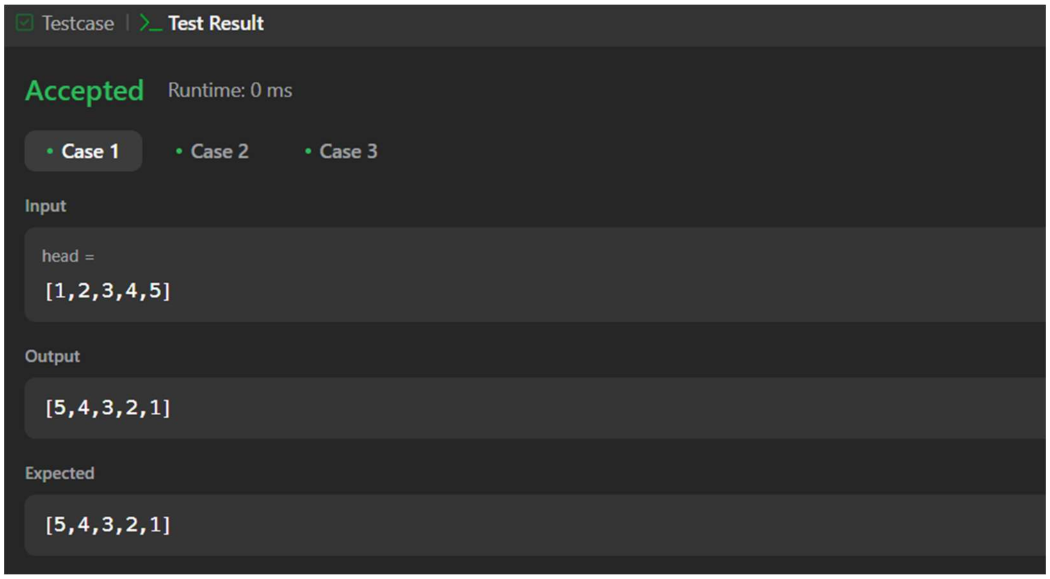
```
Accepted    Runtime: 0 ms

Input
head =
[1,1,2]

Output
[1,2]

Expected
[1,2]
```

CODE: 1.2.2

```
import java.util.*;
class ListNode {
    int val;
    ListNode next;

    ListNode(int x) {
        val = x;
        next = null;
    }

    // Deserialize a string like "[1,2,3,4,5]" into a linked list
    public static ListNode deserialize(String data) {
        data = data.replaceAll("[\\[\\]]", ""); // Remove brackets
        if (data.isEmpty()) return null;

        String[] values = data.split(",");
        ListNode head = new ListNode(Integer.parseInt(values[0]));
        ListNode current = head;

        for (int i = 1; i < values.length; i++) {
            current.next = new ListNode(Integer.parseInt(values[i].trim()));
            current = current.next;
        }

        return head;
    }

    public static void printList(ListNode head) {
        while (head != null) {
            System.out.print(head.val + " ");
            head = head.next;
        }
        System.out.println();
    }
}
```

```
class Solution {
    public ListNode reverseList(ListNode head) {
        ListNode prev = null;
        ListNode current = head;
        while (current != null) {
            ListNode next = current.next;
            current.next = prev;
            prev = current;
            current = next;
        }
        return prev;
    }
    public static void main(String[] args) {
        String input = "[1,2,3,4,5]"; // Example input format
        ListNode head = ListNode.deserialize(input);

        Solution sol = new Solution();
        ListNode reversedHead = sol.reverseList(head);
        System.out.print("Reversed List: ");
        ListNode.printList(reversedHead);
    }}
```

OUTPUT:



Accepted Runtime: 0 ms

Input

head =

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

Output

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

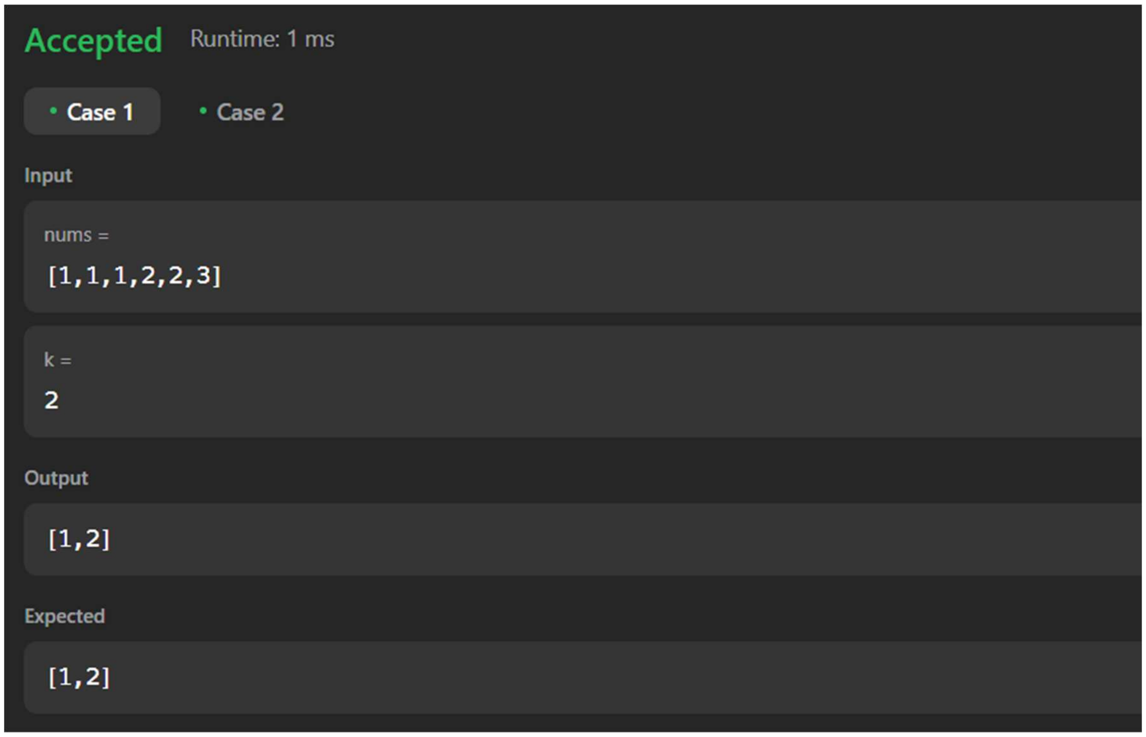
Expected

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

CODE: 1.2.3

```java
import java.util.*;

class Solution {
    public int[] topKFrequent(int[] nums, int k) {
        Map<Integer, Integer> freqMap = new HashMap<>();
        for (int num : nums) {
            freqMap.put(num, freqMap.getOrDefault(num, 0) + 1);
        }

        List<Integer>[] buckets = new List[nums.length + 1];
        for (int key : freqMap.keySet()) {
            int freq = freqMap.get(key);
            if (buckets[freq] == null) {
                buckets[freq] = new ArrayList<>();
            }
            buckets[freq].add(key);
        }

        List<Integer> result = new ArrayList<>();
        for (int i = buckets.length - 1; i >= 0 && result.size() < k; i--) {
            if (buckets[i] != null) {
                result.addAll(buckets[i]);
            }
        }

        return result.stream().mapToInt(i -> i).toArray();
    }

    public static void main(String[] args) {
        int[] nums = {1, 1, 1, 2, 2, 3};
        int k = 2;

        Solution sol = new Solution();
        int[] result = sol.topKFrequent(nums, k);
        System.out.println("Top " + k + " frequent elements: " +
Arrays.toString(result));
    }
}
```

OUTPUT:



```
Accepted   Runtime: 1 ms
Case 1   Case 2
Input
nums =
[1,1,1,2,2,3]
k =
2
Output
[1,2]
Expected
[1,2]
```

5.Learning Outcomes:

- Ability to use hash maps for frequency counting.
- Learnt to traverse a sorted linked list and remove duplicate nodes efficiently.
- Learn how to reverse a singly linked list iteratively and recursively.
- Understanding of bucket sort for optimized retrieval.
- Improved problem-solving skills for handling large datasets efficiently.