



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## WORKSHEET 2

**Student Name:** Hardik Jhamb

**UID:** 22BCS16504

**Branch:** BE-CSE

**Section/Group:** 22BCS\_NTPP-602-A

**Semester:** 6<sup>th</sup>

**Date of Performance:**

**Subject Name:** AP LAB - II

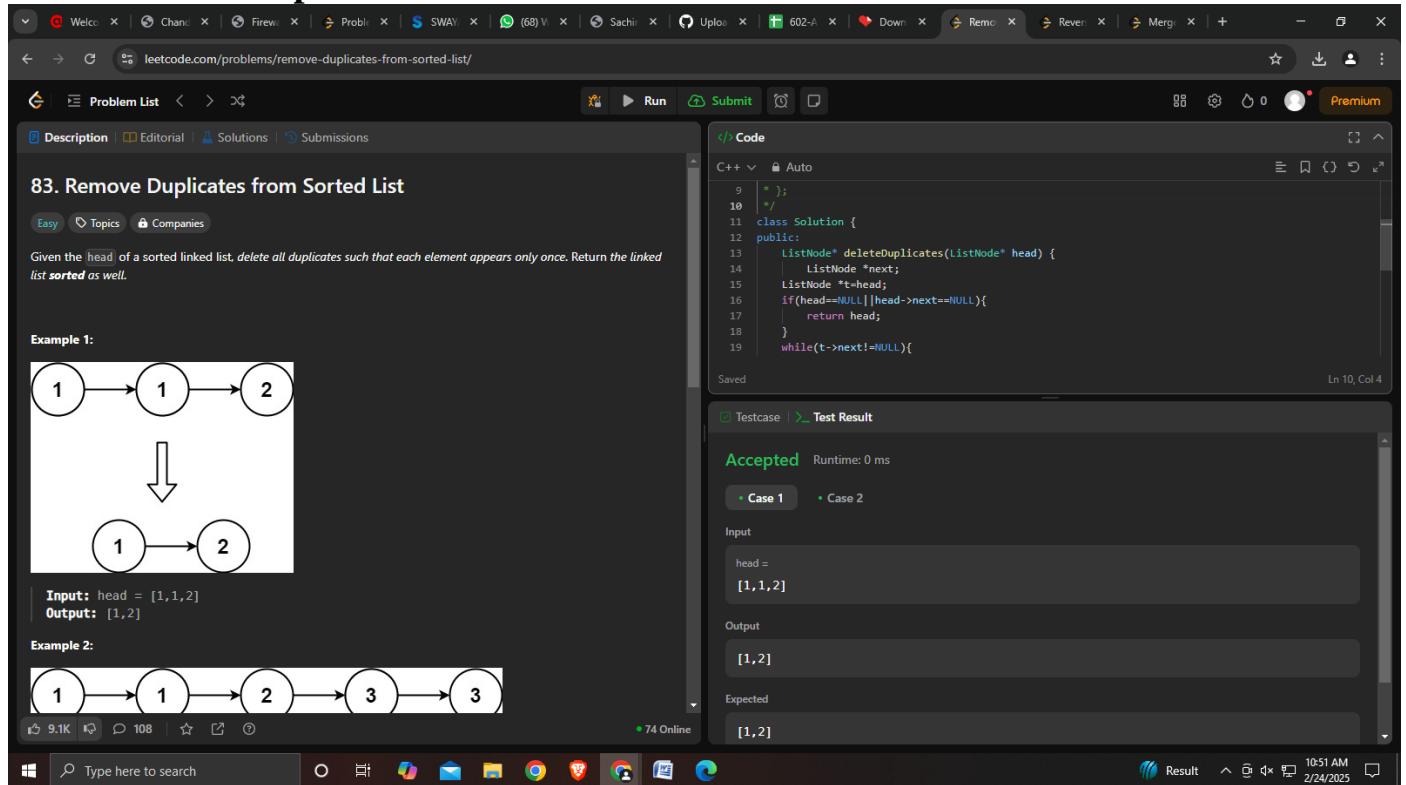
**Subject Code:** 22CSP-351

- 1. Aim:** Remove duplicates from a sorted list  
Reverse a linked list  
Merge two sorted linked lists

### **2. Source Code:**

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
        ListNode *next;
        ListNode *t=head;
        if(head==NULL||head->next==NULL){
```

### 3. Screenshots of outputs:



**83. Remove Duplicates from Sorted List**

Easy Topics Companies

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.

**Example 1:**

Input: head = [1,1,2]  
Output: [1,2]

**Example 2:**

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

```

class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
        ListNode *next;
        ListNode *t=head;
        if(head==NULL||head->next==NULL){
            return head;
        }
        while(t->next!=NULL){

```

Accepted Runtime: 0 ms

Case 1 Case 2

Input

head = [1,1,2]

Output

[1,2]

Expected

[1,2]

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

### Source Code:

```

class Solution {
public:
    ListNode* reverseList(ListNode* head) {
        // Initialize pointers
        ListNode* prev = nullptr; // Previous node starts as NULL
        ListNode* next = nullptr; // Next node
        ListNode* curr = head;    // Current node starts at the head

        // Traverse the list
        while (curr != nullptr) {
            // Save the next node
            next = curr->next;

```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## Screenshots of outputs:

The screenshot displays the LeetCode interface for the problem "206. Reverse Linked List". The problem description states: "Given the head of a singly linked list, reverse the list, and return the reversed list." Example 1 shows a linked list with nodes 1, 2, 3, 4, 5 being reversed to 5, 4, 3, 2, 1. Example 2 shows a linked list with nodes 1, 2 being reversed to 2, 1. The input for Example 1 is head = [1,2,3,4,5] and the output is [5,4,3,2,1].

The C++ code provided is as follows:

```
1 class Solution {
2 public:
3     ListNode* reverseList(ListNode* head) {
4         // Initialize pointers
5         ListNode* prev = nullptr; // Previous node starts as NULL
6         ListNode* next = nullptr; // Next node
7         ListNode* curr = head;    // Current node starts at the head
8
9         // Traverse the list
10        while (curr != nullptr) {
11            // Save the next node
```

The test result shows the solution is "Accepted" with a runtime of 0 ms. The input is head = [1,2,3,4,5] and the output is [5,4,3,2,1].



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

**Aim:** 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*.

## Source Code:

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
        ListNode* Dummy = new ListNode(0); // farzi Node
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## 4. Screenshots of outputs:

The screenshot displays a web browser window with the LeetCode problem "21. Merge Two Sorted Lists" open. The problem description states: "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." An example is provided: list1 = [1,2,4] and list2 = [1,3,4], resulting in a merged list [1,1,2,3,4,4].

The solution is implemented in C++ using a recursive approach. The code defines a ListNode struct and a recursive function mergeTwoLists that compares the values of the heads of the two lists and recursively merges the remaining nodes.

```
1 /**
2  * Definition for singly-linked list.
3  * struct ListNode {
4  *     int val;
5  *     ListNode *next;
6  *     ListNode() : val(0), next(nullptr) {}
7  *     ListNode(int x) : val(x), next(nullptr) {}
8  *     ListNode(int x, ListNode *next) : val(x), next(next) {}
9  * };
10 */
11 class Solution {
12 public:
13     ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
14         if (list1 == nullptr) return list2;
15         if (list2 == nullptr) return list1;
16         if (list1->val < list2->val) {
17             list1->next = mergeTwoLists(list1->next, list2);
18             return list1;
19         } else {
20             list2->next = mergeTwoLists(list1, list2->next);
21             return list2;
22         }
23     }
24 };
```

The test result shows the solution is "Accepted" with a runtime of 0 ms. The input lists are list1 = [1,2,4] and list2 = [1,3,4], and the output is the merged list [1,1,2,3,4,4].