**📝 Infinite Champions Programme – Day 9 (Assignment Sheet)**

**📌 Instructions  
• Deadline: Submit your solutions by 7th October, 2025, EOD.  
• Platform: Test your solutions on LeetCode  
• Collaboration: Discussing concepts is encouraged, but all code must be your own.**

1. [**Remove Duplicates from Sorted List (LeetCode #83)**](https://leetcode.com/problems/remove-duplicates-from-sorted-list/)  
   • **Problem:** Given the head of a sorted linked list, delete all duplicates such that each element appears only once.  
   • **Objective:** Traverse and adjust next pointers to skip duplicates.  
   • **YouTube Solution (Java):** [Remove Duplicates from Sorted List – Java Solution](https://www.youtube.com/watch?v=Rt4k6L1X5pU)

class Solution {

    public ListNode deleteDuplicates(ListNode head) {

        ListNode curr=head;

        while(curr!=null && curr.next!=null){

            if(curr.val==curr.next.val){

                curr.next=curr.next.next;

            }

            else{

                curr=curr.next;

            }

        }

        return head;

    }

}

1. [**Palindrome Linked List (LeetCode #234)**](https://leetcode.com/problems/palindrome-linked-list/)  
   • **Problem:** Given the head of a singly linked list, determine if it is a palindrome.  
   • **Objective:** Use slow-fast pointers, reverse second half, and compare both halves.  
   • **YouTube Solution (Java):** [Palindrome Linked List – Java Solution](https://www.youtube.com/watch?v=yOzXms1J6Nk)

class Solution {

    public ListNode findMid(ListNode head){

        ListNode slow=head;

        ListNode fast=head;

        while(fast!=null && fast.next!=null){

            slow=slow.next;

            fast=fast.next.next;

        }

        return slow;

    }

    public boolean isPalindrome(ListNode head) {

        if(head==null && head.next==null){

            return true;

        }

        ListNode midNode=findMid(head);

        ListNode prev=null;

        ListNode curr=midNode;

        ListNode next;

        while(curr!=null){

            next=curr.next;

            curr.next=prev;

            prev=curr;

            curr=next;

        }

        ListNode right=prev;

        ListNode left=head;

        while(right!=null){

            if(left.val!=right.val){

                return false;

            }

            left=left.next;

            right=right.next;

        }

        return true;

    }

}

1. [**Reorder List (LeetCode #143)**](https://leetcode.com/problems/reorder-list/)  
   • **Problem:** Reorder a linked list from L0 → L1 → … → Ln-1 → Ln into L0 → Ln → L1 → Ln-1 → L2 → Ln-2 → …  
   • **Objective:** Find middle, reverse second half, then merge alternately.  
   • **YouTube Solution (Java):** [Reorder List – Java Solution](https://www.youtube.com/watch?v=S5bfdUTrKLM)

class Solution {

    public void reorderList(ListNode head) {

        if (head == null || head.next == null){

            return;

        }

        ListNode slow = head;

        ListNode fast = head;

        while (fast != null && fast.next != null && fast.next.next != null) {

            slow = slow.next;

            fast = fast.next.next;

        }

        ListNode a= reverse(slow.next);

        slow.next = null;

        ListNode b = head;

        while (a != null) {

            ListNode tmp1 = b.next;

            ListNode tmp2 = a.next;

            b.next = a;

            a.next = tmp1;

            b = tmp1;

            a = tmp2;

        }

    }

    public ListNode reverse(ListNode head) {

        ListNode prev = null;

        ListNode curr = head;

        while (curr != null) {

            ListNode next = curr.next;

            curr.next = prev;

            prev = curr;

            curr = next;

        }

        return prev;

    }

}

**📚 Submission Checklist  
• Time and space complexity analysis for each solution.  
• Test cases demonstrating the correctness of your solutions.**