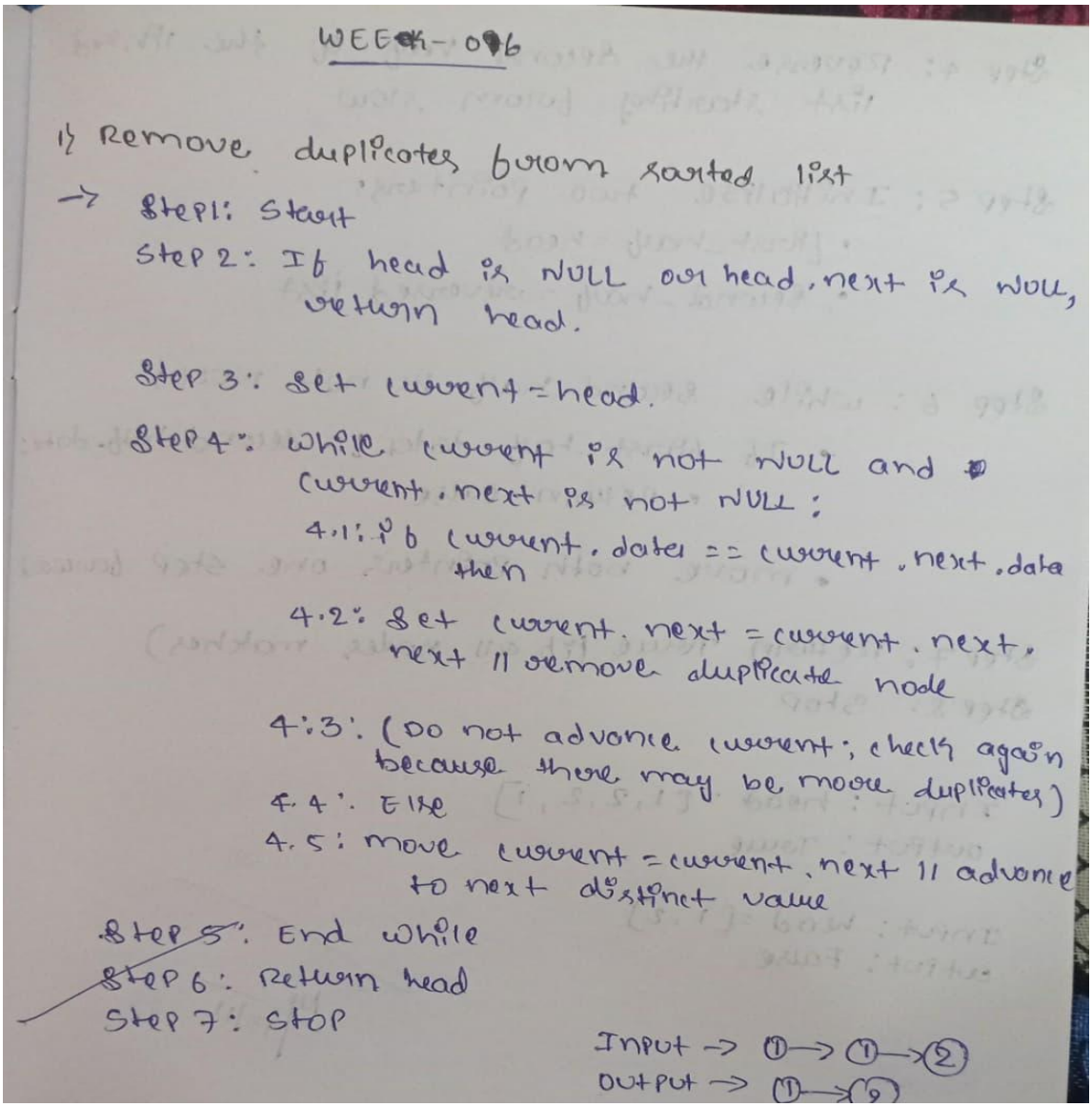
WEEK – 6

1. Remove duplicates from linked list

Algorithm:



WEEK-06

1) Remove duplicates from sorted list

→ Step1: Start
Step 2: If head is NULL or head.next is NULL, return head.
Step 3: Set current = head.
Step 4: While current is not NULL and current.next is not NULL:
4.1: If current.data == current.next.data then
4.2: Set current.next = current.next.next // remove duplicate node
4.3: (Do not advance current; check again because there may be more duplicates)
4.4: Else
4.5: move current = current.next // advance to next distinct value
Step 5: End while
Step 6: Return head
Step 7: STOP

Input → (1) → (1) → (2)
Output → (1) → (2)

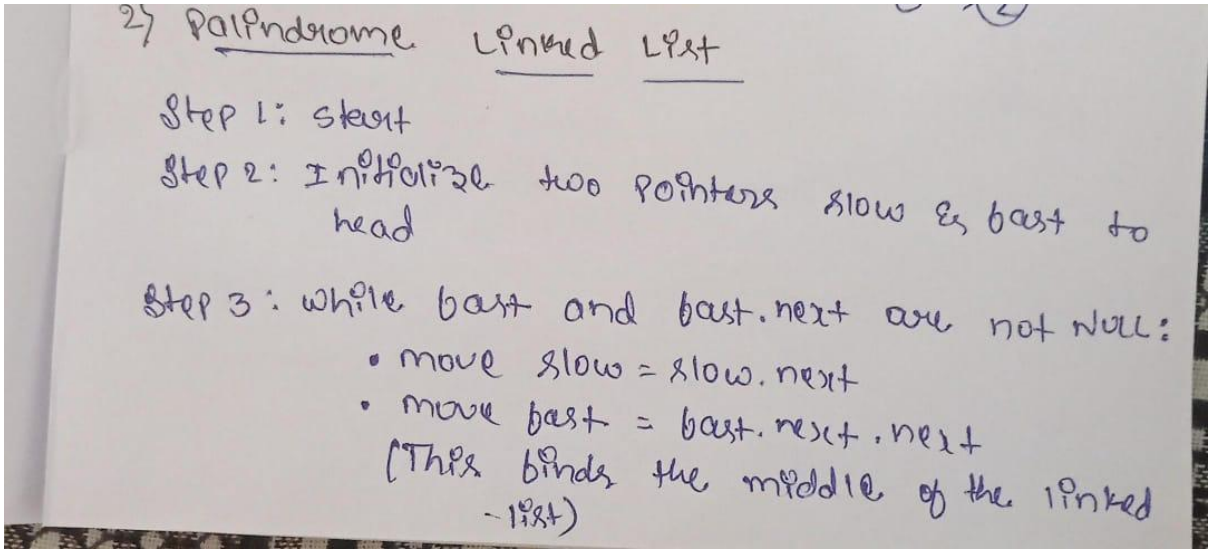
Leet Code:

C Auto

```c
/* Definition for singly-linked list.
* struct ListNode {
*     int val;
*     struct ListNode *next;
* };
*/
struct ListNode* deleteDuplicates(struct ListNode* head) {
    if (head == NULL) return NULL;

    struct ListNode* current = head;

    while (current != NULL && current->next != NULL) {
        if (current->val == current->next->val) {
            // Skip the duplicate node
            struct ListNode* temp = current->next;
            current->next = current->next->next;
            free(temp); // free memory of duplicate node (optional on LeetCode)
        } else {
            current = current->next;
        }
    }

    return head;
}
```

2. Palindrome Linked List

Algorithm:



Step 4: Reverse the second half of the linked list starting from slow

Step 5: Initialize two pointers:
- first_half = head
- second_half = reversed list

Step 6: while second_half is not NULL:
- If first_half.data != second_half.data:
  - Return false
- move both pointers one step forward

Step 7: Return True (if all nodes matches)
Step 8: Stop

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

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

13/11

Leet Code:

C Auto

```c
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */
#include <stdbool.h>
struct ListNode* reverseList(struct ListNode* head) {
    struct ListNode* prev = NULL;
    struct ListNode* curr = head;
    struct ListNode* next = NULL;

    while (curr != NULL) {
        next = curr->next;
        curr->next = prev;
        prev = curr;
        curr = next;
    }
    return prev;
}
bool isPalindrome(struct ListNode* head) {
    if (head == NULL || head->next == NULL)
        return true;

    struct ListNode* slow = head;
    struct ListNode* fast = head;
    while (fast != NULL && fast->next != NULL) {
```

```c
    struct ListNode* slow = head;
    struct ListNode* fast = head;
    while (fast != NULL && fast->next != NULL) {
        slow = slow->next;
        fast = fast->next->next;
    }
    struct ListNode* secondHalf = reverseList(slow);
    struct ListNode* firstHalf = head;
    struct ListNode* temp = secondHalf;
    while (secondHalf != NULL) {
        if (firstHalf->val != secondHalf->val)
            return false;
        firstHalf = firstHalf->next;
        secondHalf = secondHalf->next;
    }
    return true;
}
```