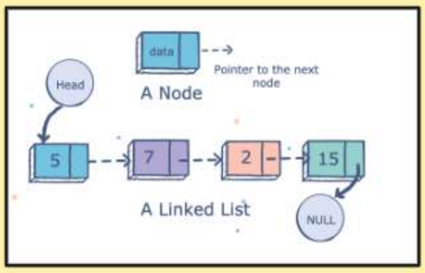
Remove Duplicates from Sorted List II (Leetcode-82)

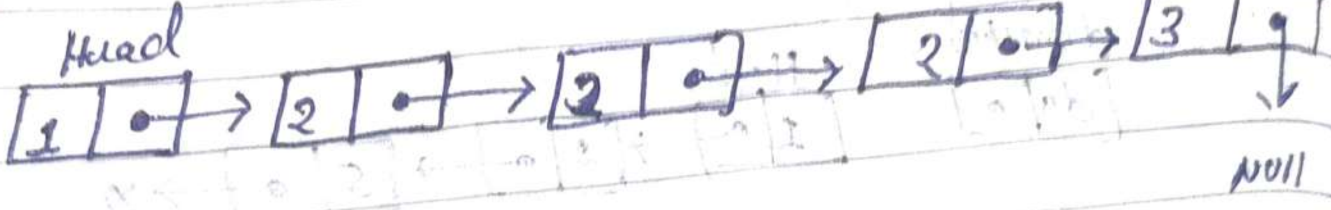
@manojofficialmj



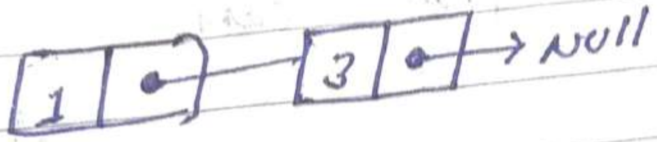
amazon Microsoft Meta ByteDance

Leetcode - 82
Remove Duplicate From
SORTED List - II

Input

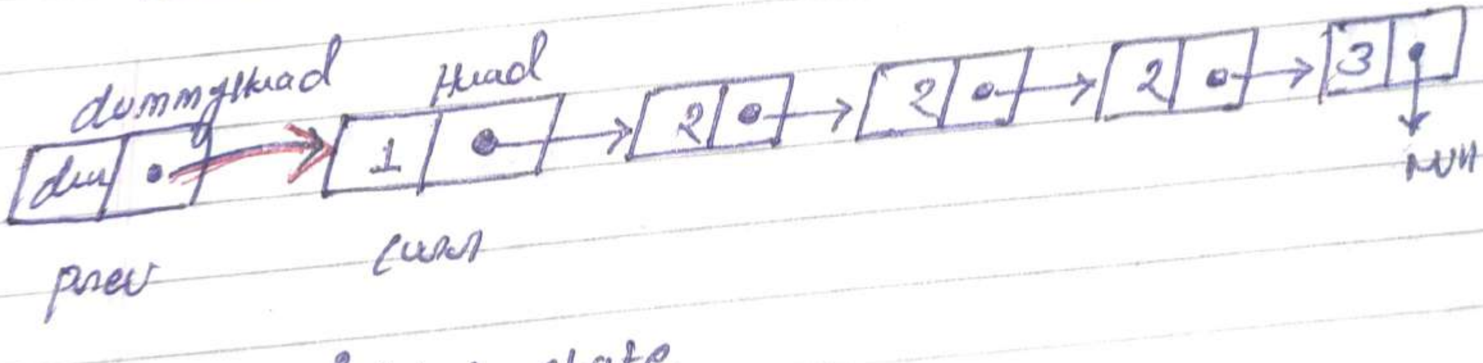


output



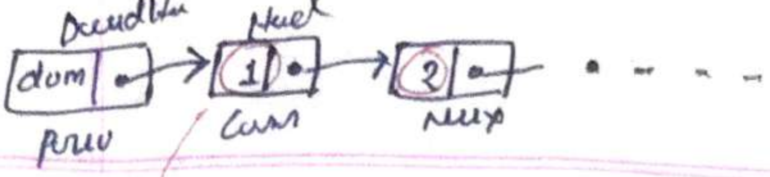
T.C. = O(N)
S.C. = O(1)

DRY RUN

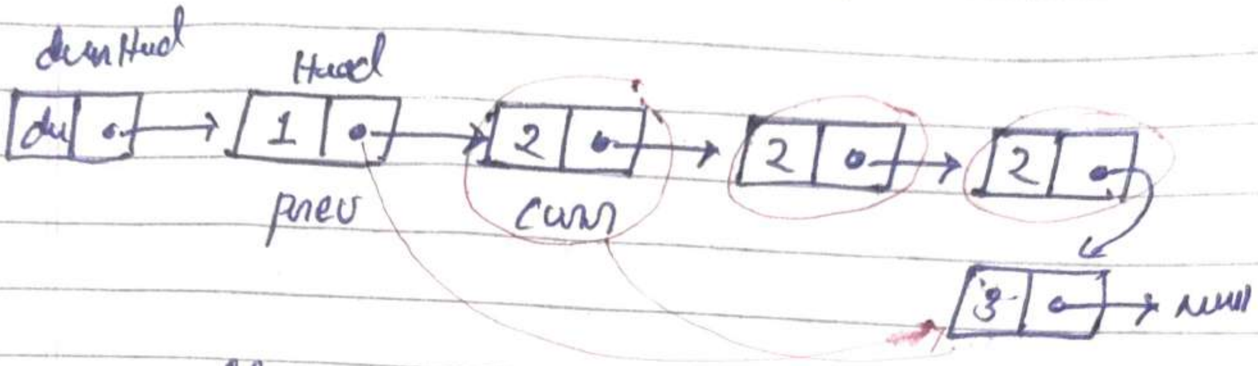


initial state

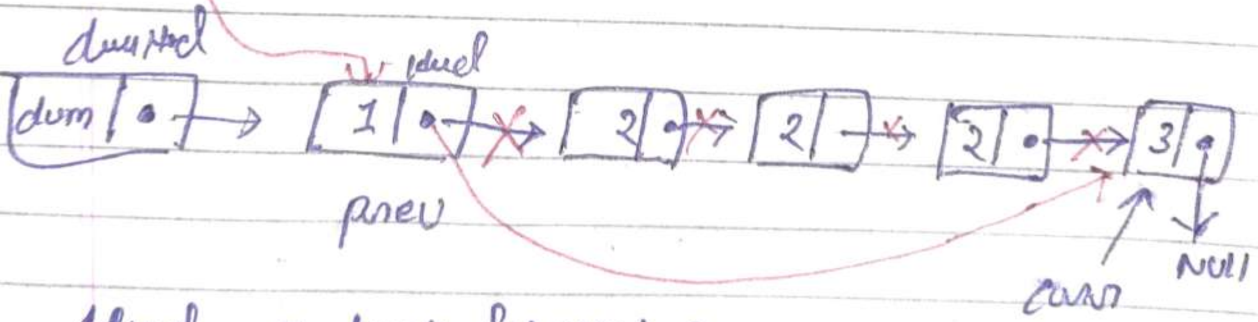
ListNode* dummyHead = Head;
ListNode* prev = dummHead;
ListNode* curr = Head;
dummyHead → next = Head;



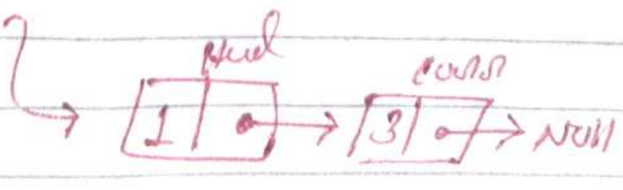
```
if (curr→val != curr→next→val)
        1   !=   2 ✓
   → prev = curr;
     curr = curr→next;
```

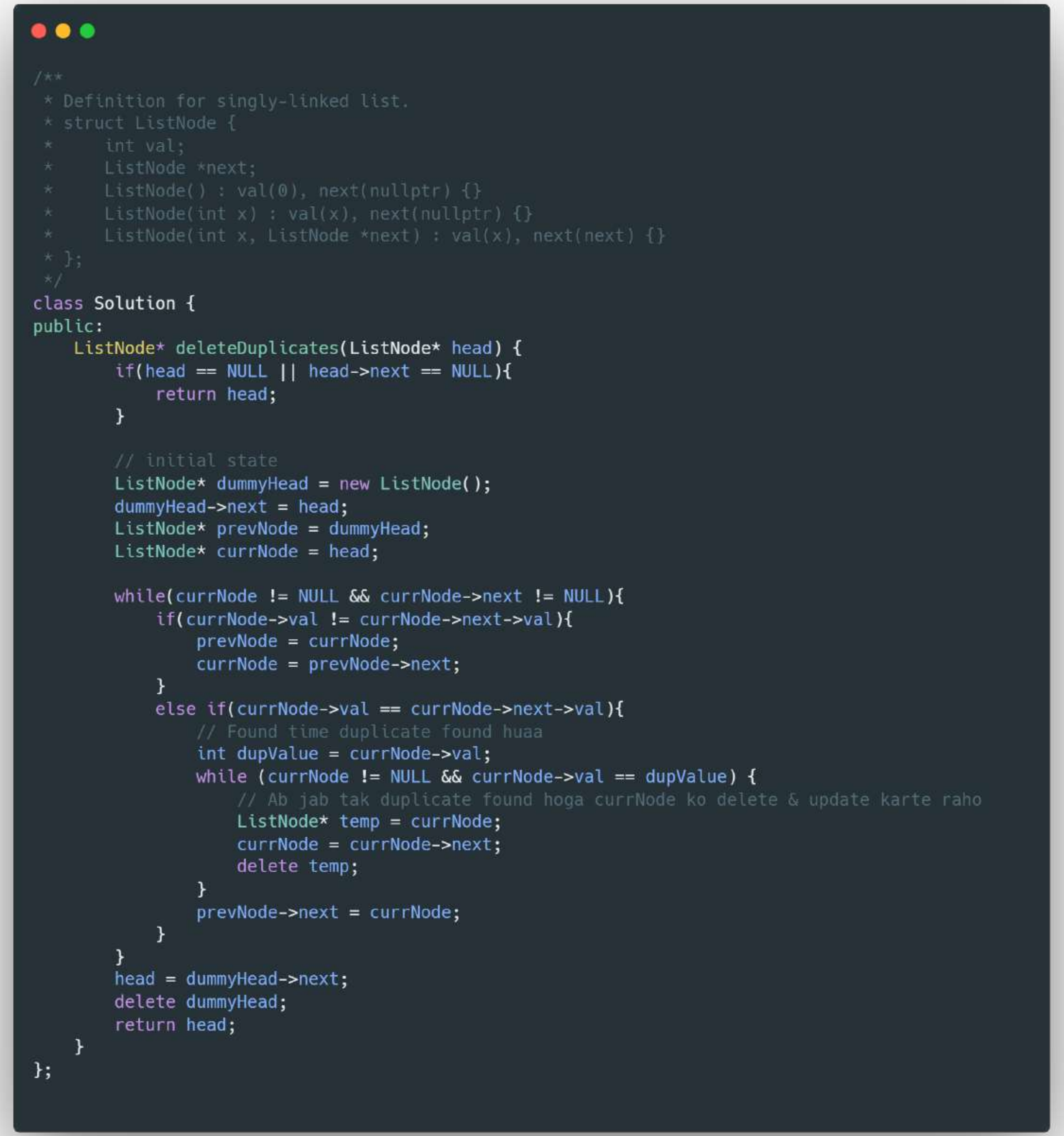


```
if (curr→val == curr→next→val) {
        2   ==   2 ✓
   int dupVal = curr→val;
   while (curr→val == dupval) {
        curr~~Node~~ = curr~~Node~~→next;
   }
   prev→next = curr~~Node~~;
}
```



```
head = dumHead→next;
delete dHead;
return head;
```





```cpp
/**
 * 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) {
        if(head == NULL || head->next == NULL){
            return head;
        }

        // initial state
        ListNode* dummyHead = new ListNode();
        dummyHead->next = head;
        ListNode* prevNode = dummyHead;
        ListNode* currNode = head;

        while(currNode != NULL && currNode->next != NULL){
            if(currNode->val != currNode->next->val){
                prevNode = currNode;
                currNode = prevNode->next;
            }
            else if(currNode->val == currNode->next->val){
                // Found time duplicate found huaa
                int dupValue = currNode->val;
                while (currNode != NULL && currNode->val == dupValue) {
                    // Ab jab tak duplicate found hoga currNode ko delete & update karte raho
                    ListNode* temp = currNode;
                    currNode = currNode->next;
                    delete temp;
                }
                prevNode->next = currNode;
            }
        }
        head = dummyHead->next;
        delete dummyHead;
        return head;
    }
};
```