**Problem Name:** Remove Duplicates from Sorted list II

**Topics:** Linked list, Two Pointers

**Companies:** Facebook, Microsoft, Amazon, Google, Apple, Bloomberg, Adobe, ByteDance

**Level:** Medium

**Language:** C++

**Problem Statement**: Given the head of a sorted linked list, delete all nodes that have duplicate numbers, leaving only distinct numbers from the original list. Return the linked list ***sorted*** as well.

**Input Format:**

First line of the input contains integer n (size of linked list)

Second line contain n space separated integer list values.

Ex:

5

1 2 3 3 5

**Output Format:** Print linked list after removing all duplicate nodes. For above input, output should be

1 2 5

**Constraints:**

* The number of nodes in the list is in the range [0, 300].
* -100 <= Node.val <= 100
* The list is guaranteed to be **sorted** in ascending order.

**Examples:**

**Input:** head = [1,2,3,3,4,4,5]

**Output:** [1,2,5]

**Solution:**

**Explanation:**

1. Basically store all the repeated elements in a set
2. Delete all those elements

**Code:**

#include <bits/stdc++.h>

using namespace std;

class ListNode

{

    public:

        int val;

        ListNode\* next;

        ListNode(int a){

            val = a;

            next = NULL;

        }

};

void insertNode(ListNode\* &head,int val) {

    ListNode\* newNode = new ListNode(val);

    if(head == NULL) {

        head = newNode;

        return;

    }

    ListNode\* temp = head;

    while(temp->next != NULL)

     temp = temp->next;

    temp->next = newNode;

    return;

}

void printList(ListNode \*node)

{

    while (node!=NULL)

    {

        cout<<node->val<<" ";

        node = node->next;

    }

}

ListNode\* deleteNode(ListNode\* head, int element){

    if(!head) return NULL;

    ListNode\* curr = head;

    ListNode\* prev = NULL;

    while(curr){

        if(head && head->val == element) head = head->next;

        if(curr && curr->val == element && prev){

            prev->next = curr->next;

        } else {

            prev = curr;

        }

        curr = prev->next;

    }

    return head;

}

ListNode\* deleteDuplicates(ListNode\* head) {

    if(!head) return NULL;

    ListNode\* curr = head;

    unordered\_set<int> s;

    while(curr && curr->next){

        if(curr->val == curr->next->val){

            s.insert(curr->val);

        }

        curr = curr->next;

    }

    for(auto x: s){

        head = deleteNode(head, x);

    }

    return head;

}

int main()

{

    ListNode\* a = NULL;

    ListNode\* res = NULL;

    int n, temp;

    cin>>n;

    while(n--){

        cin>>temp;

        insertNode(a, temp);

    }

    res = deleteDuplicates(a);

    printList(res);

    return 0;

}

**Time Complexity**: O(N2)

**Space Complexity:** O(N)

**Optimized Solution:**

**Explanation:** we want to delete all the duplicate ones including the copy. We can use the dummy node and the pre-pointer to jump over all the duplicate nodes.  
The Trap is that you may ignore that when we meet the no-duplicate numbers, we should do different op based the previous states. Just like state-machine.  
At last but not least important, we should delete the duplicate number occurs at the end.

**Code:**

#include <bits/stdc++.h>

using namespace std;

class ListNode

{

    public:

        int val;

        ListNode\* next;

        ListNode(int a){

            val = a;

            next = NULL;

        }

};

void insertNode(ListNode\* &head,int val) {

    ListNode\* newNode = new ListNode(val);

    if(head == NULL) {

        head = newNode;

        return;

    }

    ListNode\* temp = head;

    while(temp->next != NULL)

     temp = temp->next;

    temp->next = newNode;

    return;

}

void printList(ListNode \*node)

{

    while (node!=NULL)

    {

        cout<<node->val<<" ";

        node = node->next;

    }

}

ListNode\* deleteDuplicates(ListNode\* head) {

    if(!head)

        return NULL;

    ListNode\* dummy=new ListNode(INT\_MAX);

    dummy->next=head;

    ListNode\* pre=dummy, \*cur=head, \*next=head->next;

    bool flag=false;

    while(next){

        if(next->val==cur->val){

            flag=true;

            next=next->next;

        }

        else{

            if(flag) {

                pre->next=next;

                cur=next;

                next=next->next;

            }

            else{

                pre=pre->next;

                cur=cur->next;

                next=next->next;

            }

            flag=false;

        }

    }

    //the corner cases : if the duplicate number locates at the end

    if(flag) pre->next=next;

    return dummy->next;

}

int main()

{

    ListNode\* a = NULL;

    ListNode\* res = NULL;

    int n, temp;

    cin>>n;

    while(n--){

        cin>>temp;

        insertNode(a, temp);

    }

    res = deleteDuplicates(a);

    printList(res);

    return 0;

}

**Time Complexity**: O(N)

**Space Complexity:** O(1)