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

**Topics:** Linked list, Two Pointers

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

**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:**

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

Second line contain n space separated integer list values.

Ex:

7

1 2 3 3 4 4 5

**Output Format:** Print linked list after removing all duplicate nodes.

**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]

**Brute force Solution:**

**Explanation:** Store all the repeated elements in a set. 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:

* Traverse the list from the head (or start) node.
* While traversing, compare each node with its next node.
* If the data of the next node is the same as the current node then delete the next node.
* Before we delete a node, we need to store the next pointer of the node.

**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 0;

    if(!head -> next)

        return head;

    int val = head ->val;

    ListNode \*temp = head -> next;

    if(temp -> val != val){

        head -> next = deleteDuplicates(temp);

        return head;

    }

    else{

        while(temp && temp -> val == val){

            ListNode \*p = temp;

            temp = temp->next;

            delete p;

        }

        return deleteDuplicates(temp);

    }

}

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)