**Remove duplicate element from sorted Linked List: -**

Easy Accuracy: 49.37% Submissions: 222K+ Points: 2

Given a singly linked list consisting of **N** nodes. The task is to remove duplicates (nodes with duplicate values) from the given list (if exists).  
**Note:** Try not to use extra space. The nodes are arranged in a **sorted**way.

**Example 1:**

**Input:**

LinkedList: 2->2->4->5

**Output:** 2 4 5

**Explanation:** In the given linked list

2 ->2 -> 4-> 5, only 2 occurs more

than 1 time. So we need to remove it once.

**Example 2:**

**Input:**

LinkedList: 2->2->2->2->2

**Output:** 2

**Explanation:** In the given linked list

2 ->2 ->2 ->2 ->2, 2 is the only element

and is repeated 5 times. So we need to remove  
any four 2.

**Your Task:**  
The task is to complete the function **removeDuplicates**() which takes the head of input linked list as input. The function should remove the duplicates from linked list and return the head of the linkedlist.

**Expected Time Complexity** : O(N)  
**Expected Auxilliary Space** : O(1)

**Constraints:**  
1 <= Number of nodes <= 105

**Code: -**

//{ Driver Code Starts

#include <bits/stdc++.h>

using namespace std;

struct Node {

int data;

struct Node \*next;

Node(int x) {

data = x;

next = NULL;

}

};

void print(Node \*root)

{

Node \*temp = root;

while(temp!=NULL)

{

cout<<temp->data<<" ";

temp=temp->next;

}

}

Node\* removeDuplicates(Node \*root);

int main() {

// your code goes here

int T;

cin>>T;

while(T--)

{

int K;

cin>>K;

Node \*head = NULL;

Node \*temp = head;

for(int i=0;i<K;i++){

int data;

cin>>data;

if(head==NULL)

head=temp=new Node(data);

else

{

temp->next = new Node(data);

temp=temp->next;

}

}

Node \*result = removeDuplicates(head);

print(result);

cout<<endl;

}

return 0;

}

// } Driver Code Ends

/\*

struct Node {

int data;

struct Node \*next;

Node(int x) {

data = x;

next = NULL;

}

};\*/

//Function to remove duplicates from sorted linked list.

Node \*removeDuplicates(Node \*head)

{ Node \*ptr=head, \*prev=head;

while(ptr){

if(ptr->data == prev->data){

while(ptr and ptr->data == prev->data)

ptr = ptr->next;

prev->next = ptr;

prev = ptr;

}

}

return head;

}

**T.C :- O(N)**

**S.C :- O(1)**