**83. Remove Duplicates from Sorted List**

Given the head of a sorted linked list, *delete all duplicates such that each element appears only once*. Return *the linked list****sorted****as well*.

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

**Output:** [1,2,3]

**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.

**SOLUTION:**

class Node<T> {

T? value;

Node<T>? next;

Node(this.value);

}

class LinkedList<T> {

Node<T>? head;

LinkedList();

bool get isEmpty => head == null;

void add(T value) {

final newNode = Node<T>(value);

if (isEmpty) {

head = newNode;

} else {

var current = head;

while (current!.next != null) {

current = current.next;

}

current.next = newNode;

}

}

void remove(T value) {

if (isEmpty) return;

if (head!.value == value) {

if (head != null) {

head = head!.next;

}

return;

}

var current = head;

while (current!.next != null) {

if (current.next!.value == value) {

if (current.next != null) {

current.next = current.next!.next;

}

return;

}

current = current.next;

}

}

void printList() {

var current = head;

while (current != null) {

print(current.value);

current = current.next;

}

}

void removeDuplicate() {

var current = head;

while (current != null && current.next != null) {

if (current.value == current.next!.value) {

current.next = current.next!.next;

} else {

current = current.next;

}

}

}

}

void main() {

final linkedList = LinkedList<int>();

linkedList.add(-100);

linkedList.add(-100);

linkedList.add(-50);

linkedList.add(-50);

linkedList.add(0);

linkedList.add(0);

linkedList.add(50);

linkedList.add(50);

linkedList.add(100);

linkedList.add(100);

linkedList.printList();

linkedList.removeDuplicate();

print("AFTER DELETIION: ");

linkedList.printList();