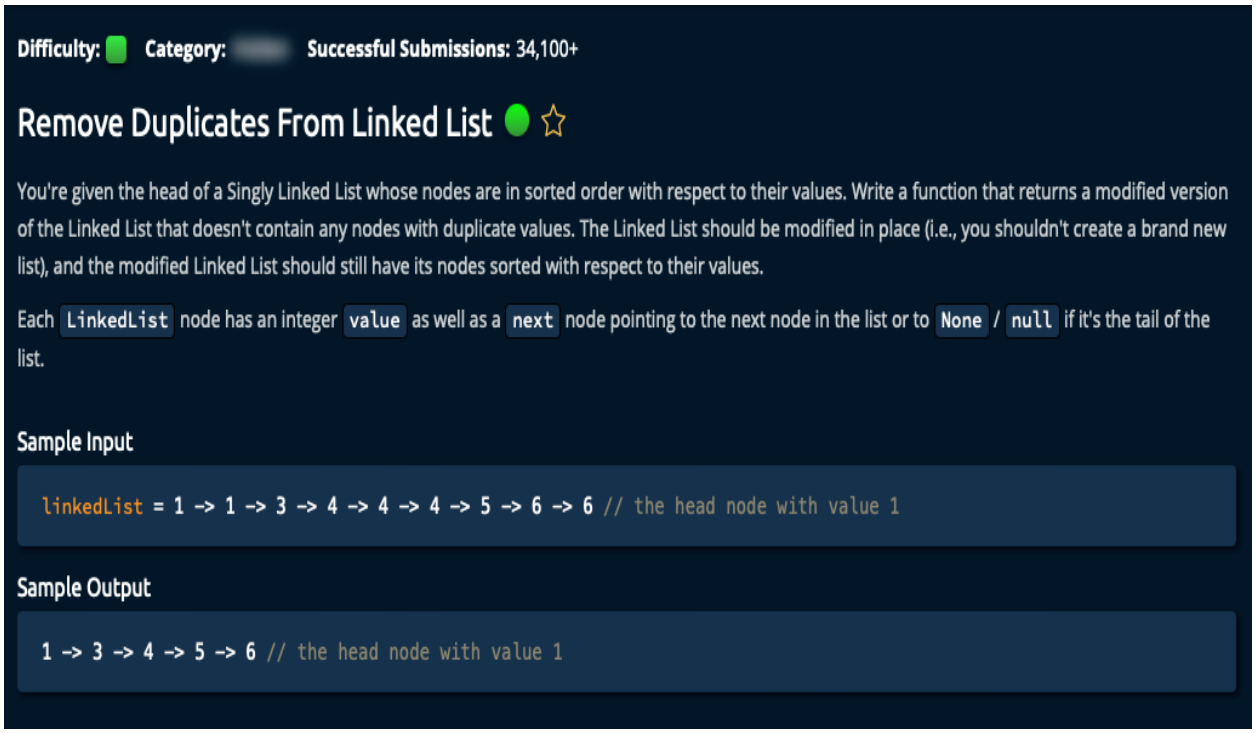
Code Challenge #16 Remove Duplicates from Linked List (Easy)



Difficulty: **Category:** **Successful Submissions:** 34,100+

Remove Duplicates From Linked List

You're given the head of a Singly Linked List whose nodes are in sorted order with respect to their values. Write a function that returns a modified version of the Linked List that doesn't contain any nodes with duplicate values. The Linked List should be modified in place (i.e., you shouldn't create a brand new list), and the modified Linked List should still have its nodes sorted with respect to their values.

Each `LinkedList` node has an integer `value` as well as a `next` node pointing to the next node in the list or to `None` / `null` if it's the tail of the list.

Sample Input

```
linkedList = 1 -> 1 -> 3 -> 4 -> 4 -> 4 -> 5 -> 6 -> 6 // the head node with value 1
```

Sample Output

```
1 -> 3 -> 4 -> 5 -> 6 // the head node with value 1
```

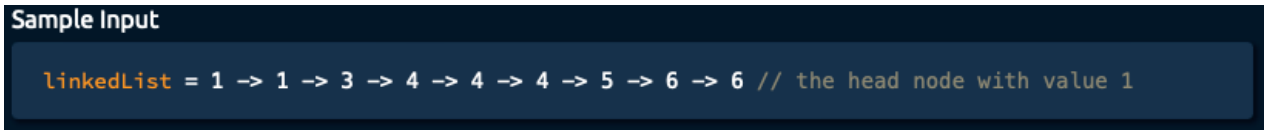
Solution #1

```
1. // This is an input class. Do not edit.
2. class LinkedList {
3.   constructor(value) {
4.     this.value = value;
5.     this.next = null;
6.   }
7. }
8.
9. function removeDuplicatesFromLinkedList(linkedList) {
10.    let currentNode = linkedList;
11.         while (currentNode !== null) {
12.                let nextDistinctNode = currentNode.next;
13.                while (nextDistinctNode  !== null &&
   nextDistinctNode.value === currentNode.value) {
14.                       nextDistinctNode = nextDistinctNode.next;
15.                }
16.                currentNode.next =nextDistinctNode;
17.                currentNode = nextDistinctNode;
18.         }
```

```
19.    return linkedList;
20. }
21.
```

Explanation

The solution to this problem is based on manipulating pointers in a Linked List. The Linked List in this problem is integers in ascending order. This means that the duplicate numbers are next to each other. See below.



This makes finding the duplicates much easier as we compare values next to each other and remove the links between the linked lists if they match. We have the LinkedList pointer using .next point to the next item and check if the Current Node matches the next Node. We first create a function called removeDuplicatesFromLinkedList which takes an argument called LinkedList. Inside the function we have a let variable called currentNode which is equal to the linkedList. We then use a while loop that runs as long as currentNode != null. As long as this is the case we have a let variable called nextDistinctNode which is point to the next value using currentNode.next. We then have a second while loop which runs as long as nextDistinctNode !== null && nextDistinctNode.value === currentNode.Value. If those conditions are true we have the nextDistinctNode point to nextDistinctNode.next. Outside of the second while loop we assign currentNode.next to nextDistinct and currentNode is equal to nextDistinctNode. We then finally return the LinkedList. In a LinkedList we change pointers aka next so that we no longer have the nodes we want. This code runs in O(n) time.