[Click on the logo to find **Problem Statement**]

Intuition

This code aims to remove all occurrences of a specified value (`val`) from a linked list in-place. It utilizes a dummy node to handle edge cases with the head of the list. The algorithm iterates through the list with two pointers (`prev` and `head`). For each node, if its value matches `val` , it is skipped; otherwise, it is included in the modified list by updating the `prev` pointer. After traversal, the last included node's `next` is set to `None` for proper termination. The modified list, excluding nodes with the specified value, is returned.

Approach

1. **Dummy Node Initialization:**
    - Create a dummy node and set its `next` pointer to the original head of the linked list. This dummy node simplifies the handling of edge cases where the head of the list needs to be modified.
2. **Traversal with Two Pointers:**
    - Initialize two pointers, `prev` and `head` , both initially pointing to the dummy node.
    - Iterate through the list using the `head` pointer.
3. **Value Check and Modification:**
    - For each node, check if its value is equal to the target value, `val` .
    - If the value matches, skip this node in the modified list.
    - If the value does not match, update the `prev` pointer to link to the current node, effectively including it in the modified list.
4. **Termination and Final Adjustment:**
    - After traversing the entire list, set the `next` of the last included node (pointed by `prev`) to `None` to ensure the correct termination of the modified list.
5. **Return Modified List:**
    - Return the `next` of the dummy node, which points to the modified list.

Complexity

- Time complexity: O(n)
- Space complexity: O(1)

Code

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
  def removeElements(self, head: ListNode, val: int) -> ListNode:
    dummy = ListNode(0, head)
    prev = dummy

    while head:
      if head.val != val:
        prev.next = head
        prev = prev.next
      head = head.next
    prev.next = None

    return dummy.next
```

If you want to see more solutions to coding problems, you can visit: