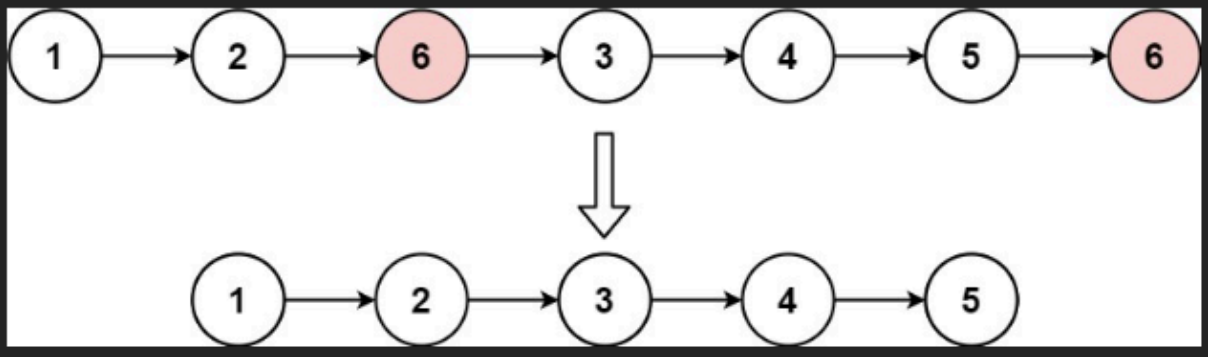
203. Remove Linked List Elements

Easy | Topics | Companies

Given the `head` of a linked list and an integer `val`, remove all the nodes of the linked list that has `Node.val == val`, and return *the new head*.

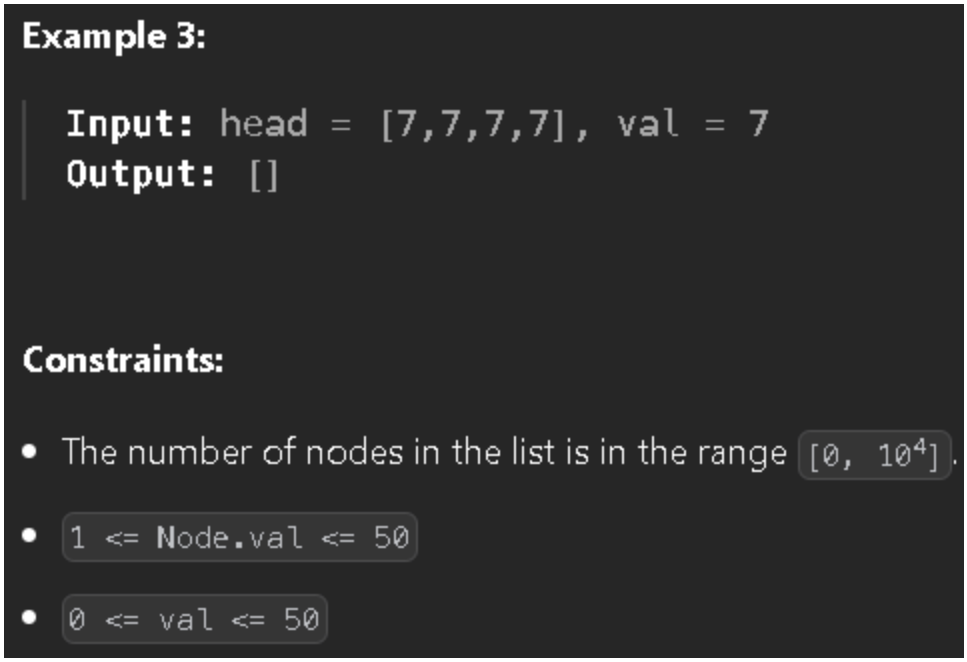
Example 1:




```
Input: head = [1,2,6,3,4,5,6], val = 6
Output: [1,2,3,4,5]
```

Example 2:

```
Input: head = [], val = 1
Output: []
```



Python:

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next

from typing import Optional

class Solution:
    def removeElements(self, head: Optional[ListNode], val: int) -> Optional[ListNode]:
        # Create a dummy node to simplify deletion cases (especially head removals)
        dummy = ListNode(0)
        dummy.next = head
        current = dummy

        # Traverse the list
        while current.next:
            if current.next.val == val:
                # Skip the node with value == val
                current.next = current.next.next
            else:
                # Move forward
                current = current.next

        # Return new head (skipping dummy)
```

```
        return dummy.next
```

JavaScript:

```
/**
 * Definition for singly-linked list.
 * function ListNode(val, next) {
 *     this.val = (val===undefined ? 0 : val)
 *     this.next = (next===undefined ? null : next)
 * }
 */
/**
 * @param {ListNode} head
 * @param {number} val
 * @return {ListNode}
 */
var removeElements = function(head, val) {
    // Create a dummy node before head
    let dummy = new ListNode(0);
    dummy.next = head;

    let current = dummy;

    // Traverse the list
    while (current.next !== null) {
        if (current.next.val === val) {
            // Skip the node
            current.next = current.next.next;
        } else {
            current = current.next;
        }
    }

    return dummy.next; // New head
};
```

Java:

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
```

```
 *     ListNode(int val, ListNode next) { this.val = val; this.next = next; }
 * }
 */
class Solution {
    public ListNode removeElements(ListNode head, int val) {
        // Create a dummy node pointing to head
        ListNode dummy = new ListNode(0);
        dummy.next = head;

        // Use current pointer to traverse the list
        ListNode current = dummy;

        while (current.next != null) {
            if (current.next.val == val) {
                // Skip the node with value == val
                current.next = current.next.next;
            } else {
                // Move forward
                current = current.next;
            }
        }

        // Return the new head (dummy.next)
        return dummy.next;
    }
}
```