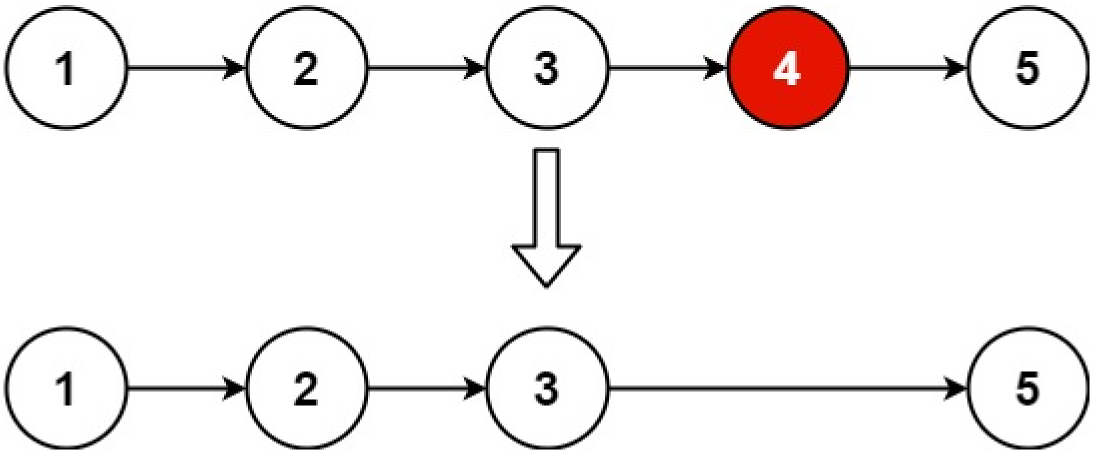
19. Remove Nth Node From End of List

Given the `head` of a linked list, remove the `n<sup>th</sup>` node from the end of the list and return its head.

Example 1:



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

Example 2:

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

Example 3:

```
Input: head = [1,2], n = 1
Output: [1]
```

Constraints:

- The number of nodes in the list is `sz`.
- `1 <= sz <= 30`
- `0 <= Node.val <= 100`
- `1 <= n <= sz`

Follow up: Could you do this in one pass?

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

```
#         self.val = val
#         self.next = next
class Solution:
    def removeNthFromEnd(self, head: Optional[ListNode], n: int) ->
Optional[ListNode]:
        if head is None or head.next is None:
            return None
        fast = head
        slow = head
        prev = None
        count = 0
        while fast.next!=None:
            if count<n-1:
                fast = fast.next
                count = count +1
            else:
                prev = slow
                slow = slow.next
                fast = fast.next
        if prev is None:
            head = slow.next
            slow.next = None
        else:
            prev.next = slow.next
            slow.next = None
        return head
```