# 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]:

# Create a dummy node to handle edge cases, especially when removing the head node.

dummy = ListNode(0, head)

slow = dummy

fast = dummy

# Move the fast pointer 'n' steps ahead.

# This creates a gap of 'n' nodes between fast and slow.

for \_ in range(n):

fast = fast.next

# Move both pointers until the fast pointer reaches the end of the list.

# When fast.next is None, fast is at the last node.

# At this point, slow will be at the node \*before\* the one to be removed.

while fast.next:

slow = slow.next

fast = fast.next

# Remove the nth node from the end by updating slow.next.

slow.next = slow.next.next

# Return the head of the modified list, which is dummy.next.

return dummy.next