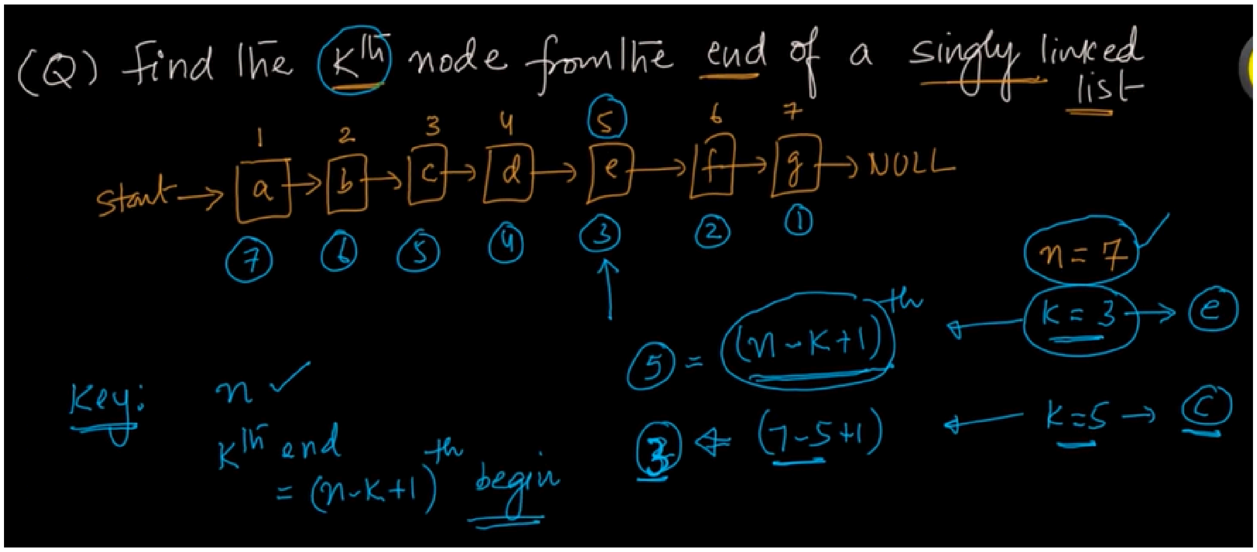
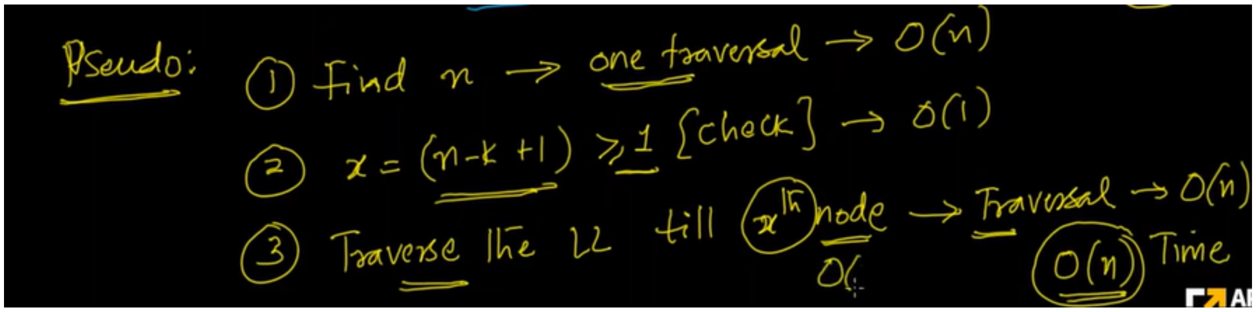
Find the kth node from end of SLL

1. 

2. Here the basic approach would be…count the number of nodes in SLL and return (n-k+1)th node
3. Pseudo code for this approach:

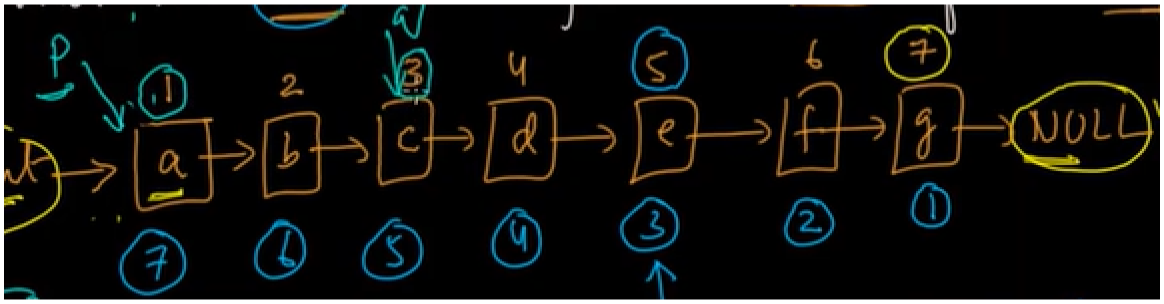


```
Pseudo:
  ① Find n → one traversal → O(n)
  ② x = (n-k+1) ≥ 1 {check} → O(1)
  ③ Traverse the LL till x^th node → Traversal → O(n)
     O(            O(n) Time
```

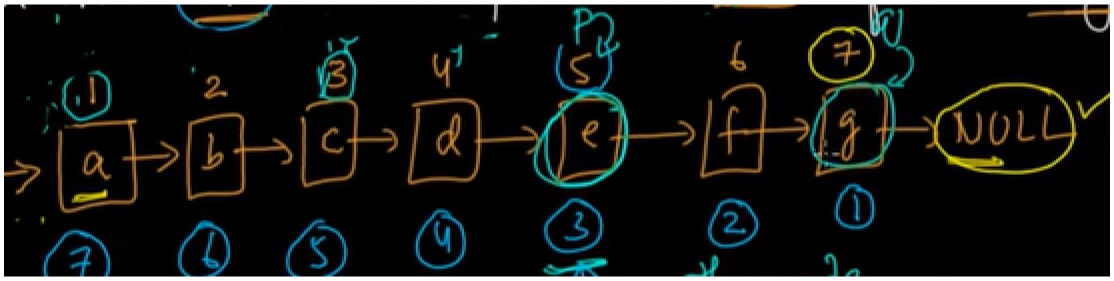
4. Here the time complexity is O(n) but we are taking two traversal

More Optimized (one Traversal)

1. Lets take 2 pointers and initialize them to head
2. Now move the 2nd pointer to k times

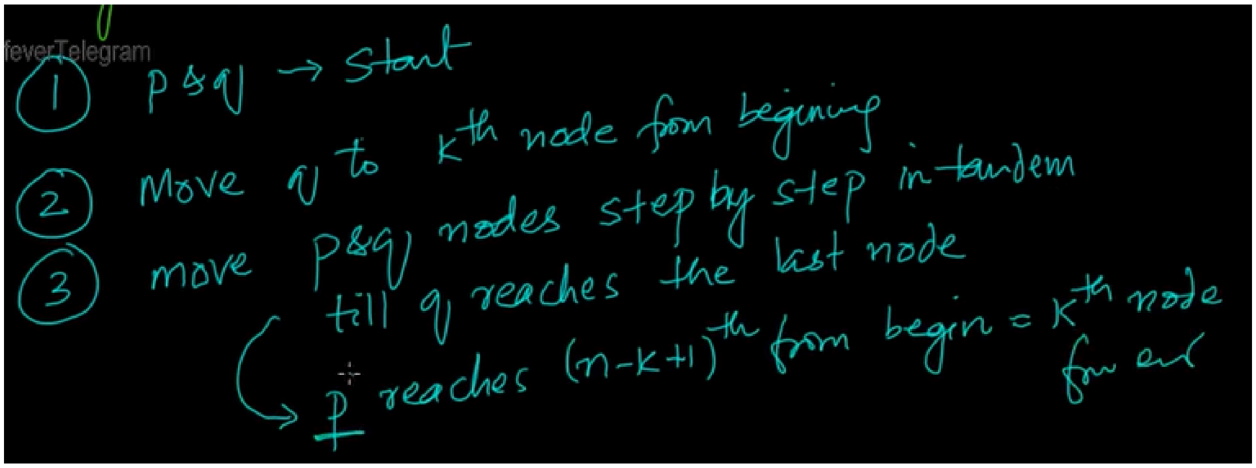


3. Now we move these both pointers until q reaches the null..

4. 

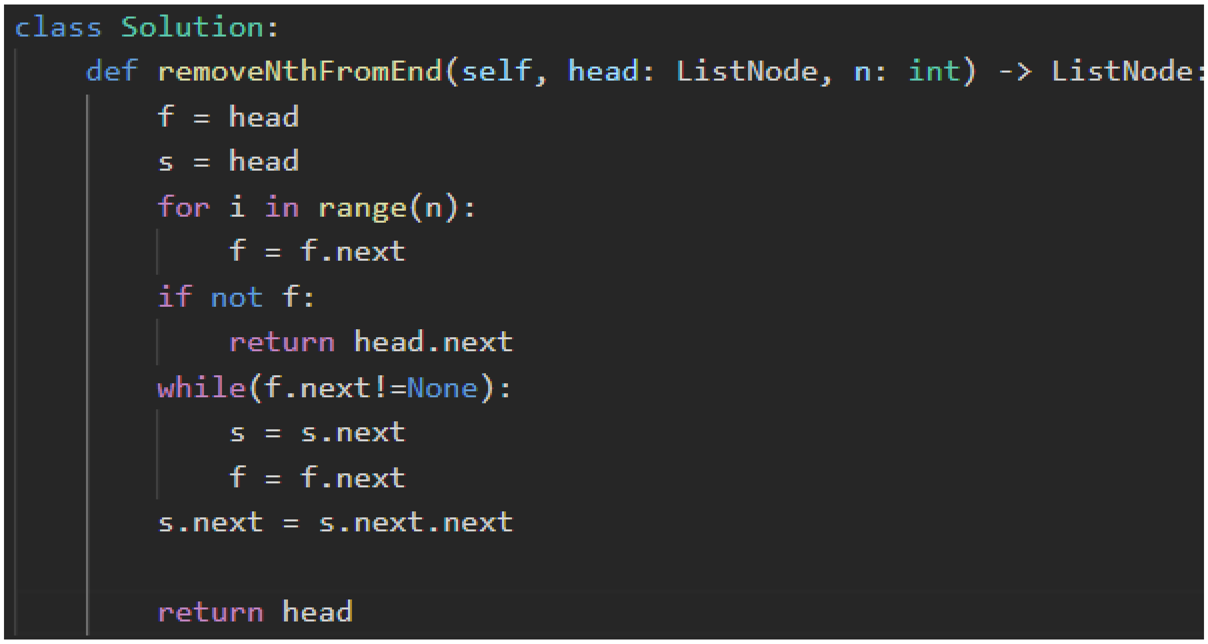
here P gives the nth node from the end

5. Pseudocode for this approach


feverTelegram

```
(1) p & q → start
(2) Move q to kth node from begining
(3) move p&q nodes step by step in tandem
    till q reaches the last node
    ↳ p reaches (n-k+1)th from begin = kth node from end
```

Code for 19. Remove Nth Node From End of List

1. 

```
class Solution:
    def removeNthFromEnd(self, head: ListNode, n: int) -> ListNode:
        f = head
        s = head
        for i in range(n):
            f = f.next
        if not f:
            return head.next
        while(f.next!=None):
            s = s.next
            f = f.next
        s.next = s.next.next

        return head
```

```
# Edge case: If q becomes None (list has fewer than n nodes)
if not q:
    return head.next  # Remove the head node and return the new head
```

2.