A one pass solution can be done using pointers. Move one pointer **fast** --> **n+1** places forward, to maintain a gap of n between the two pointers and then move both at the same speed. Finally, when the fast pointer reaches the end, the slow pointer will be **n+1** places behind - just the right spot for it to be able to skip the next node.

Since the question gives that **n** is valid, not too many checks have to be put in place. Otherwise, this would be necessary.

**public** ListNode removeNthFromEnd(ListNode head, **int** n) {

ListNode start = **new** ListNode(0);

ListNode slow = start, fast = start;

slow.**next** = head;

//Move fast in front so that the gap between slow and fast becomes n

**for**(**int** i=1; i<=n+1; i++) {

fast = fast.**next**;

}

//Move fast to the end, maintaining the gap

**while**(fast != **null**) {

slow = slow.**next**;

fast = fast.**next**;

}

//Skip the desired node

slow.**next** = slow.**next**.**next**;

**return** start.**next**;

}