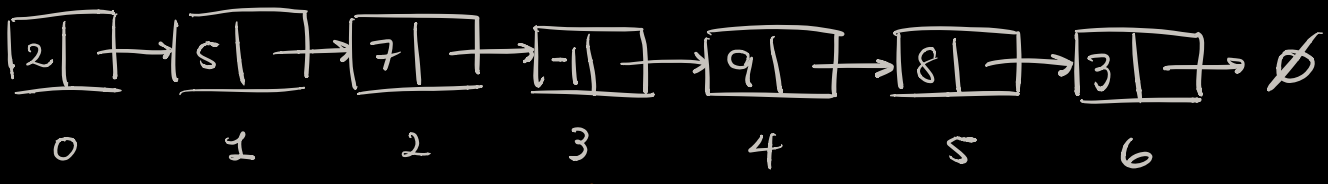
Return k^{th} to Last: Implement an algorithm to find the k^{th} to last element of a singly linked list.

Example:



if k = 2, return 8

if k = 4, return -1

- since we are finding the k^{th} to last element, the algorithm should involve tracking the last element.

Iterative Solutions:

- size is unknown.

1) A brute force solution would be to iterate through the linked list & return the size.

In our example, size = 7.

Given k, we can subtract k-1 from the size.

if k = 2,

$$size - (k-1) = 7 - (2-1) = 7 - 1 = 6$$

So we would iterate through the linked list again, with a new counter, and return the node when the counter hits 6.

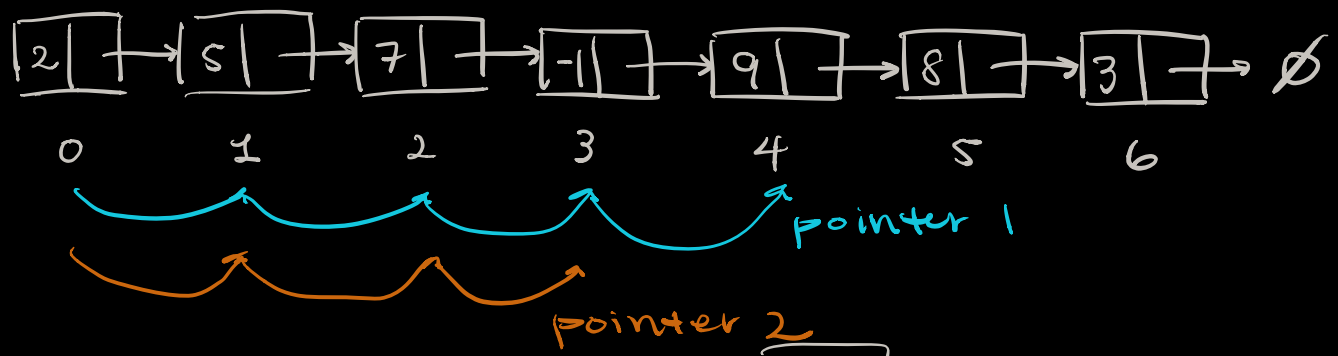
Time Complexity:

$O(2n) = O(n)$

Space Complexity:

$O(1)$, no additional data structures were used

2) But a better? solution would be to have two pointers at different positions iterate the linked list.



The second pointer being $k-1$ nodes behind.

When pointer 1 is on the last node, pointer 2 should be on the node we want to return.

Implementation:

```
getKtoLastNode(rootNode, k) {
    if (rootNode === null) {
        return null;
    }

    let counter = 0;
    let frontpointer, backpointer = rootNode;
    while (frontpointer.next !== null) {
        if(counter >== k-1) {
            backpointer = backpointer.next;
        }
        frontpointer = frontpointer.next;
        counter++;
    }

    return backpointer.data;
}
```