Problem Domain

You are given a linked list, a param, that is an integer and asked to return the node's value that is k places from the tail/end of the linked list
arguments: int, k as a param
return: node's value that is k places from the tail of the linked list
input: head →{1}→{3}→{8}→{2}→ X input arg: 0
output: 2
input: head →{1}→{3}→{8}→{2}→ X input arg: 2
output: 3
input: head →{1}→{3}→{8}→{2}→ X input arg: 6
output: exception

Algorithm

create a function that accepts an integer input and then starting at the head of a linked list, the function will traverses the list to count items in the linked list
once the count is known the compare the count w/ the integer input
if it is <= the value of the integer input is not present
if >= the input is present and the value can be returned

Tests/Edge Cases

1. tests provided via classwork:

input: head →{1}→{3}→{8}→{2}→ X input arg: 0
output: 2
input: head →{1}→{3}→{8}→{2}→ X input arg: 2
output: 3

2. exceptions provided via classwork: where k is greater than the length of the linked list

input: head →{1}→{3}→{8}→{2}→ X input arg: 6
output: exception

3. where k and the length of the list are the same

input: head →{1}→{3}→{8}→{2}→ X input arg: 3
output: 1

4. where k is a negative int

input: head →{1}→{3}→{8}→{2}→ X input arg: -1
output: exception

5. where linked list is the size of 1

input: head →{2}→ X input arg: 0
output: 2

6. "Happy path" where k is not at the end, but in the middle of the linked list

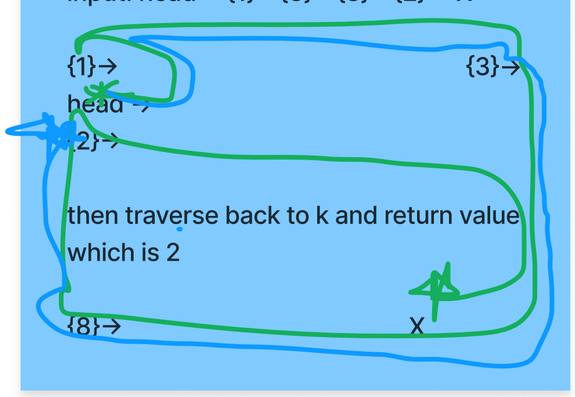
input: head →{1}→{3}→{8}→{2}→{5}→ X input arg: 3
output: 8

7. where the linked list is empty

input: [] input arg: 0
output: exception

Visualization

input arg(k): 0
input: head →{1}→{3}→{8}→{2}→ X



Big O

time: O(n) where n is the length of the linked list, constant
space: O(l) we are only required to store a few variables to traverse the list, linear

Step Through

1	input: head →{1}→{3}→{8}→{2}→ X input arg(k): 0
2	traverse the list to see the len of list, and count the num of Nodes using a while loop
2a	count variable counts and increments 1 node
2b	count variable counts and increments 2 node
2c	count variable counts and increments 3 node
2d	count variable counts and increments 4 node
3	using a for loop, set current to head of linked list, then create a sequence of values that represents the number of nodes that must be traversed from the head to reach the kth node from the end
3a	Loop moves current value to {1}
3b	Loop moves current value to {3}
3c	Loop moves current value to {8}
3d	Loop moves current value to {2}, loop concludes on this step because the value matches the input of 0, outputs value of {2}

Code

```
 def kth_from_end(self, k):
        if k < 0:
            raise TargetError("k must be a non-negative integer")

        current = self.head
        count = 0
        while current:
            count += 1
            current = current.next

        if count == 0:
            raise TargetError("Cannot get kth element from an empty list")

        if k > count:
            raise TargetError ("k is greater than or equal to the number of nodes in the linked list")

        current = self.head
        for i in range(count - k - 1):
            current = current.next

        return current.value
```