25. Reverse Nodes in K Group

Hard

Given the head of a linked list, reverse the nodes of the list k at a time, and return the modified list.

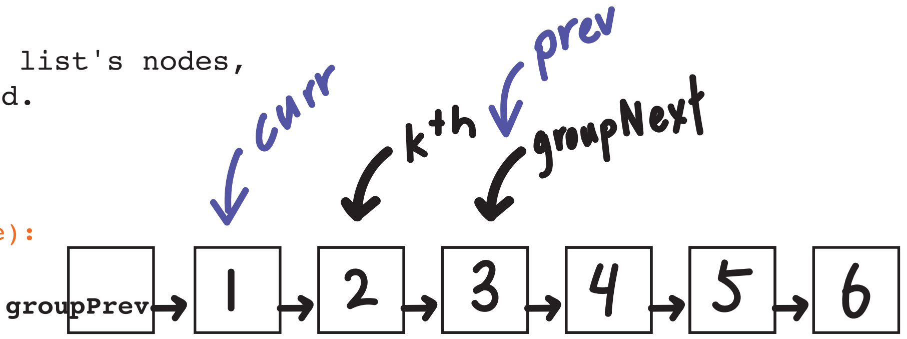
k is a positive integer and is less than or equal to the length of the linked list. If the number of nodes is not a multiple of k then left-out nodes, in the end, should remain as it is.

You may not alter the values in the list's nodes, only nodes themselves may be changed.

Example 1.
Input: head = [1,2,3,4,5], k = 2
Output: [2,1,4,3,5]

Example 2.
Input: head = [1,2,3,4,5], k = 3
Output: [3,2,1,4,5]

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
```

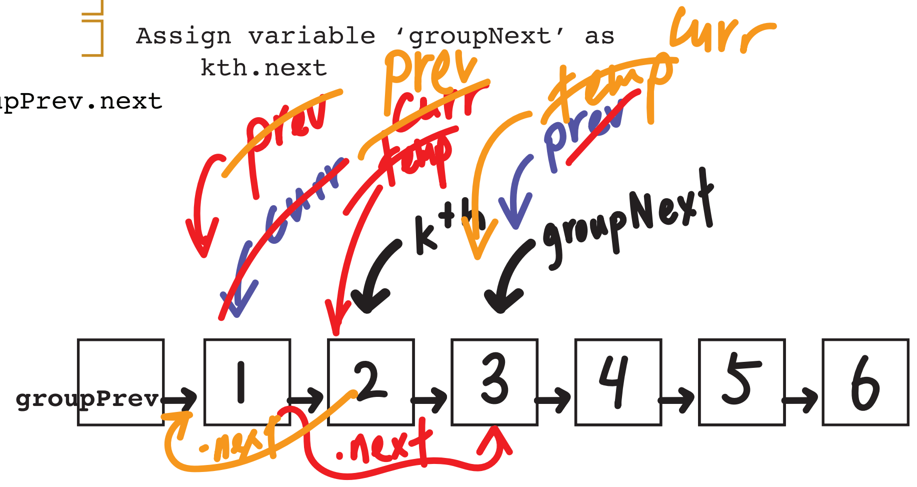


```
class Solution:
    def reverseKGroup(self, head: Optional[ListNode], k: int) -> Optional[ListNode]:
        dummy = groupPrev = ListNode(0, head)

        while True:
            kth = self.getKth(groupPrev, k)
            if not kth:
                break
            groupNext = kth.next

            prev, curr = kth.next, groupPrev.next
            while curr != groupNext:
                temp = curr.next
                curr.next = prev
                prev = curr
                curr = temp

            tmp = groupPrev.next
            groupPrev.next = kth
            groupPrev = tmp
        return dummy.next

    def getKth(self, curr, k):
        while curr and k > 0:
            curr = curr.next
            k -= 1
        return curr
```

Outputs the k'th node

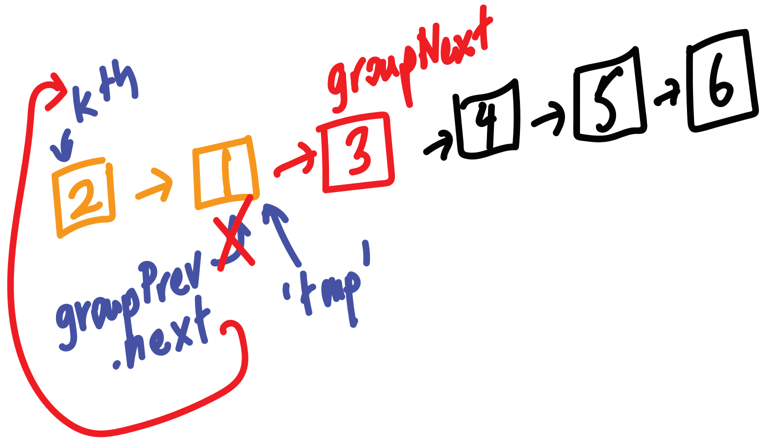
Breaks loop if the 'kth' is None

Assign variable 'groupNext' as kth.next



'getKth()' gets the kth node based on 'k' and 'curr', the current node

iterate over the linked list
until the kth node is reached.
k -= 1
curr = curr.next



kth
groupNext
2
1
3
4
5
6
groupPrev
.next
'tmp'