Problem 3: 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.

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


def reverseKGroup(head, k):
    def reverse_group(start, end):
        prev, curr = None, start
        while curr != end:
            temp = curr.next
            curr.next = prev
            prev = curr
            curr = temp
        return prev

    def find_kth_node(node, k):
        for i in range(k):
            if not node:
                return None
            node = node.next
        return node

    dummy = ListNode(0)
    dummy.next = head
    prev_group_tail = dummy
    current = head

    while current:
        group_end = find_kth_node(current, k)
```

```
        if not group_end:
            break
        next_group_head = group_end.next
        new_group_head = reverse_group(current, group_end)
        prev_group_tail.next = new_group_head
        current.next = next_group_head
        prev_group_tail = current
        current = next_group_head
    return dummy.next


def create_linked_list(lst):
    dummy = ListNode(0)
    current = dummy
    for val in lst:
        current.next = ListNode(val)
        current = current.next
    return dummy.next


def linked_list_to_list(head):
    lst = []
    current = head
    while current:
        lst.append(current.val)
        current = current.next
    return lst


head = create_linked_list([1, 2, 3, 4, 5, 6, 7, 8])
k = 3
result = reverseKGroup(head, k)
print(linked_list_to_list(result))
```


64
input
[3, 2, 1, 7, 6, 5]
...Program finished with exit code 0
Press ENTER to exit console.