1. Reverse Nodes in k-Group

Hard

Given a linked list, reverse the nodes of a linked list *k* at a time and return its 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.

**Example:**

Given this linked list: 1->2->3->4->5

For *k* = 2, you should return: 2->1->4->3->5

For *k* = 3, you should return: 3->2->1->4->5

**Note:**

* Only constant extra memory is allowed.
* You may not alter the values in the list’s nodes, only nodes itself may be changed.

**解法1**

递归求解。关于链表翻转的所有题型（直接翻转、配对翻转、组内翻转）均可使用递归求解

**解法2**

直接分组，每组组内进行翻转。

class Solution {  
public:  
 ListNode \*reverseKGroup(ListNode \*head, int k) {  
 // 容错处理  
 if(head == NULL || k < 2){  
 return head;  
 }  
 //添加头结点  
 ListNode \*dummy = new ListNode(0);  
 dummy->next = head;  
 ListNode \*pre,\*cur,\*tail;  
 pre = dummy;  
 // 分组旋转的第一个节点即旋转后的尾节点  
 tail = head;  
 // 当前节点  
 cur = head;  
 int count = 0;  
 // 统计节点个数  
 while(cur != NULL){  
 cur = cur->next;  
 count++;  
 }  
 // 旋转次数  
 int rCount = count / k;  
 // 分组旋转下标  
 int index = 0;  
 // 旋转  
 while(rCount){  
 // 分组旋转  
 // k节点只需旋转k-1节点  
 index = k-1;  
 while(index){  
 //先删除  
 cur = tail->next;  
 tail->next = cur->next;  
 //再插入  
 cur->next = pre->next;  
 pre->next = cur;  
 index--;  
 }//while  
 pre = tail;  
 tail = tail->next;  
 rCount--;  
 }//while  
 return dummy->next;  
 }  
};