K 个一组翻转链表

给你一个链表，每 *k* 个节点一组进行翻转，请你返回翻转后的链表。

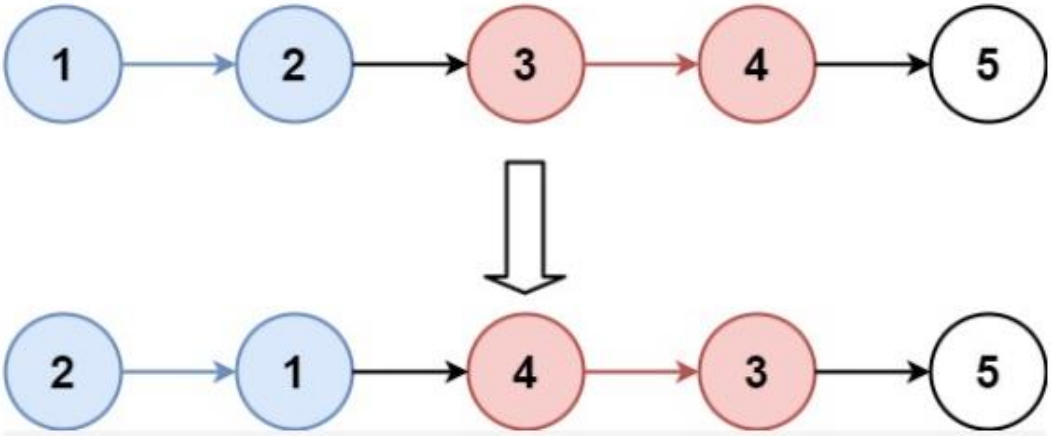
k 是一个正整数，它的值小于或等于链表的长度。

如果节点总数不是 *k* 的整数倍，那么请将最后剩余的节点保持原有顺序。

进阶：

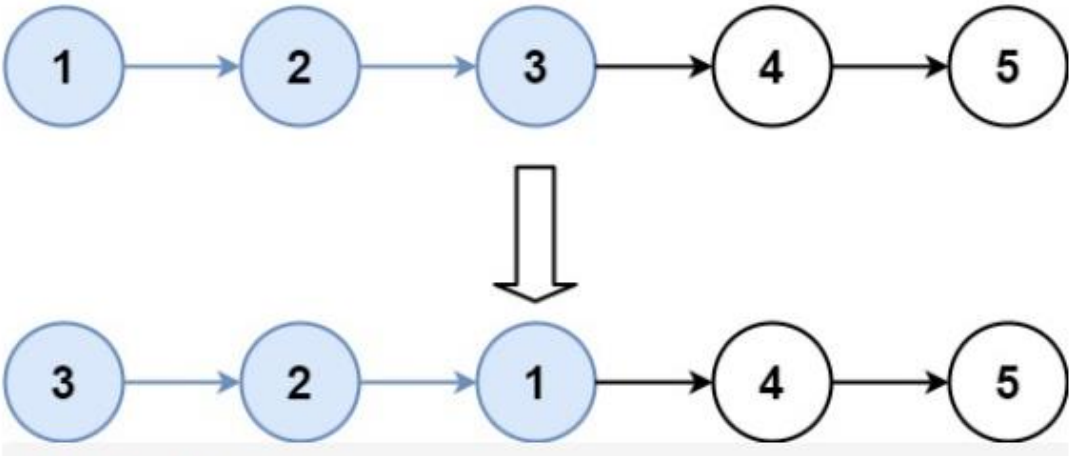
- 你可以设计一个只使用常数额外空间的算法来解决此问题吗？
- **你不能只是单纯的改变节点内部的值**，而是需要实际进行节点交换。

示例 1：




```
输入：head = [1,2,3,4,5], k = 2
输出：[2,1,4,3,5]
```

示例 2：




```
输入：head = [1,2,3,4,5], k = 3
输出：[3,2,1,4,5]
```

```cpp
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    //翻转 n 个节点链表
    ListNode* reverse(ListNode* head, int n)
    {
        ListNode* prev=nullptr;
        ListNode* cur=head;
        for(int i=0;i<n;i++)
        {
            ListNode* next=cur->next;
            cur->next=prev;
            prev=cur;
            cur=next;
        }
        return prev;
    }

    ListNode* reverseKGroup(ListNode* head, int k) {
        ListNode* nexthead=head;//表示 k 个节点后，下一个即将翻转链表的头结点
        int count=0;
        while(count<k)
        {
            if(nexthead==nullptr)
                return head;//表示后续无可翻转链表
            count++;
            nexthead=nexthead->next;//记录下一个即将翻转链表的头结点
        }
         //翻转当前 k 个节点链表
        ListNode* newhead=reverse(head,k);
        ListNode* subList=reverseKGroup(nexthead,k);//递归调用下一个翻转列表
        head->next=subList;
        return newhead;
    }
};
```