```c
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */

void reverse(struct ListNode* head, struct ListNode* tail) {
    struct ListNode* tmp;
    struct ListNode* next;
    struct ListNode* prev;

    if(head != NULL && tail != NULL)
    {
        tmp = head;
        prev = NULL;

        while(true)
        {
            next = tmp->next;

            if(prev)
            {
                tmp->next = prev;
            }

            if(tmp == tail)
            {
                break;
            }else
            {
                prev = tmp;
                tmp = next;
            }

        }

        head->next = NULL;

        tmp = tail;
    }
}

struct ListNode* reverseKGroup(struct ListNode* head, int k) {

    struct ListNode* fast;
    struct ListNode* slow;
    struct ListNode* prev;
    struct ListNode* reversed_head;
    struct ListNode* reversed_tail;
    struct ListNode* result;
    struct ListNode* result_last;
    int count;

    fast = head;
    slow = head;
    reversed_head = fast;
    count = 0;
    result = NULL;

    while(true)
    {
        slow = slow->next;
        prev = fast;
        fast = fast->next == NULL ? NULL : fast->next->next;

        count+=2;

        if(count >= k)
        {
            reversed_tail = (count == k) ? prev->next : prev;

            if(reversed_tail)
            {
                fast = reversed_tail->next;
                slow = reversed_tail->next;
                reverse(reversed_head, reversed_tail);

                if(result == NULL)
                {
                    result = reversed_tail;
                    result_last = reversed_head;
                }else
                {
                    result_last->next = reversed_tail;
                    result_last = reversed_head;
                }

                reversed_head = fast;
            }

            count = 0;
            if(fast != NULL && fast->next == NULL)
            {
                break;
            }
```

```
        }

        if(fast == NULL)
        {
            break;
        }

    }


    if(result == NULL)
    {
       result = reversed_head;
    }else
    {
       result_last->next = reversed_head;
    }

    return result;
}
```