C++方向编程题答案

答案说明：

> 大家如果对本次题目或者答案有问题，可以联系下方的出题老师答疑。
>
> **出题老师：**
>
> 选择题：鲍松山 qq：365690203
>
> 代码题1：吴都 qq：1226631755
>
> 代码题2：张晓寅 qq：3508973948

题号: 537557 反转部分单向链表

[链接]

https://www.nowcoder.com/questionTerminal/f11155006f154419b0bef6de8918aea2

[题目描述]

给定一个单链表，在链表中把第 L 个节点到第 R 个节点这一部分进行反转。

输入描述:

```
n 表示单链表的长度。val 表示单链表各个节点的值。L 表示翻转区间的左端点。R 表示翻转区间的右端点。
```

输出描述:

```
在给定的函数中返回指定链表的头指针。
```

输入

```
5
1 2 3 4 5
1 3
```

输出

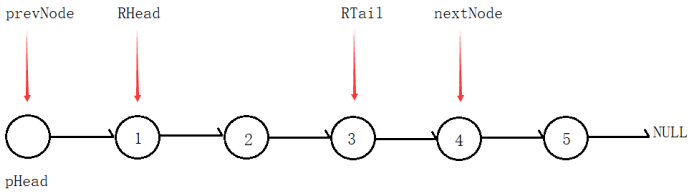
```
3 2 1 4 5
```

思路一:

> 找到需要反转部分链表的起始位置,断链反转之后,再进行恢复链表输出.

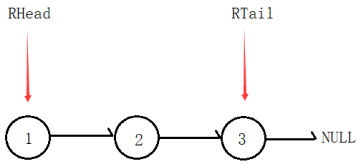
图解:

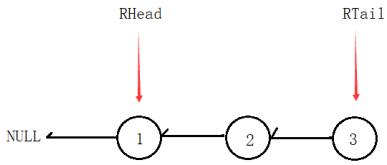
找到需要的节点：



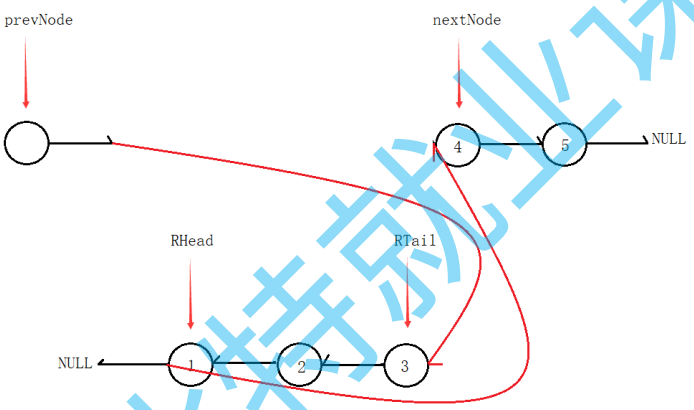
prevNode:反转部分链表的前一个节点
RHead:反转部分链表的头结点
RTail:反转部分链表的尾节点
nextNode:反转部分链表的下一个节点

断链+逆置：

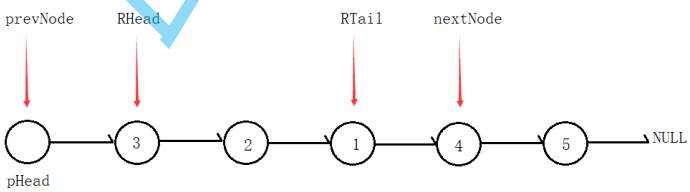




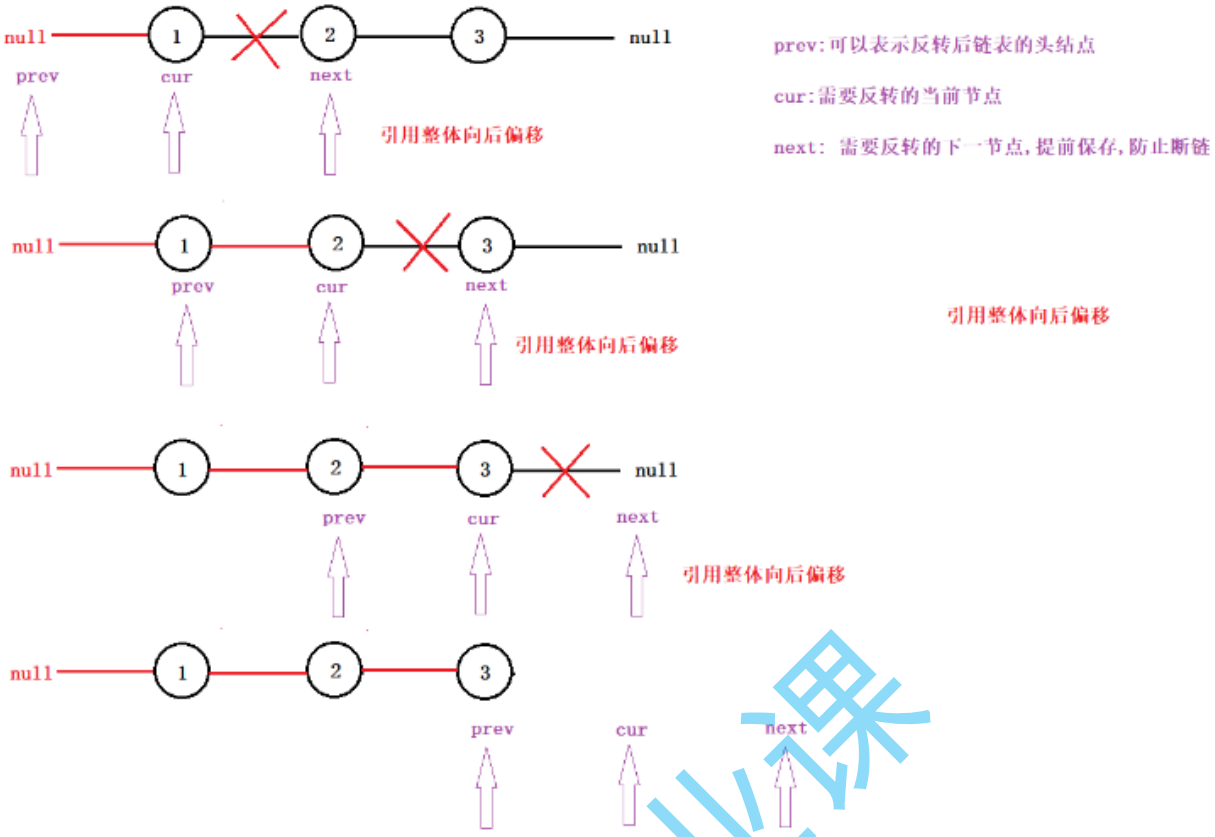
恢复链接：



逆置结果：



双指针反转链表的过程：



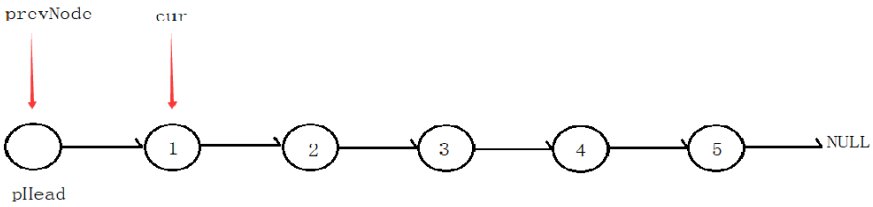

```
list_node* ReverseList(list_node* head)
{
    list_node* prev = nullptr,*cur = head;
    while(cur)
    {
        list_node* next = cur->next;
        cur->next = prev;
        prev = cur;
        cur = next;
    }
    return prev;
}

list_node * reverse_list(list_node * head, int L, int R)
{
    list_node* newHead = new list_node;
    newHead->next = head;
    list_node* prevNode = newHead,*RHead = nullptr,*RTail = nullptr,*TailNext = nullptr;
    for(int i = 0;i < L-1;i++)
    {
        prevNode = prevNode->next;
    }
    RHead = prevNode->next;
    RTail = RHead;

    for(int i = 0;i < R-L;i++)
```

```
    {
        RTail = RTail->next;
    }
    TailNext = RTail->next;
    RTail->next = nullptr;
    list_node* RNewHead = ReverseList(RHead);
    prevNode->next = RNewHead;
    RHead->next = TailNext;
    return newHead->next;
}
```

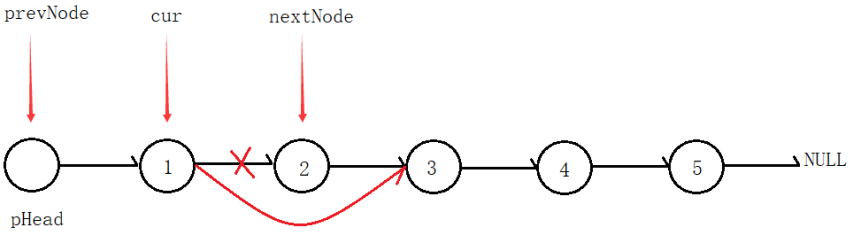
思路二:

> 思路一的弊端:假设需要反转的链表部分,占比比较大,则需要两次遍历链表来实现. (1.遍历确定反转链表的起始位置 2.遍历链表进行反转). 那是不是可以考虑一次遍历链表就解决该问题呢?
>
> 这里的思路是:采用头插的方式一次遍历解决问题

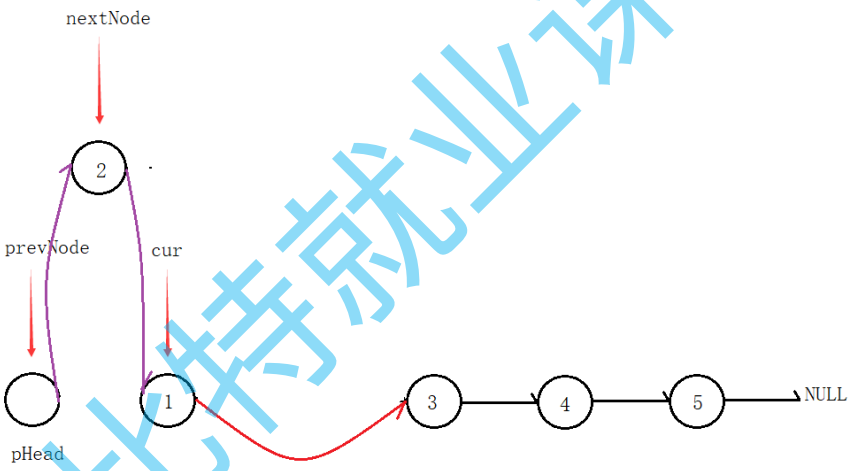
图解:



prevNode：表示反转部分链表的前一个节点
cur:表示反转链表的头结点

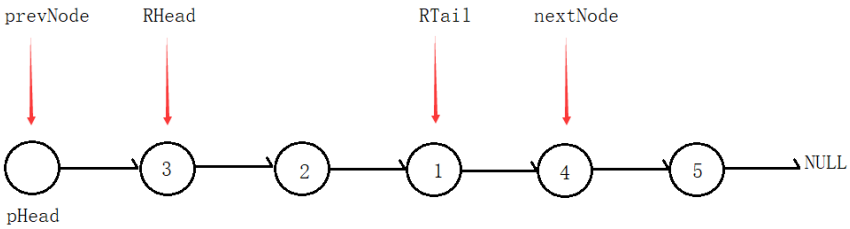


1. 先找到需要头插的节点，使用nextNode指针进行记录
2. 为了防止头插之后断链，这里先让cur的next指针指向nextNode的下一个节点，便于后续的操作



3. 然后将nextNode的next指针指向prevNode的下一个节点[注意：先进行这步操作，否则找不到prevNode的下一个节点进行链接]
4. prevNode的next指针指向nextNode指向的节点，即可完成头插
5. 重复以上操作，即可完成部分链表的逆置（重复几次呢？ 这块是R-L次，R-L就能得到反转部分链表节点个数-1，这个数字也就是需要头插的节点个数

逆置结果：




```
list_node * reverse_list(list_node * head, int L, int R)
{
    list_node* pHead = new list_node;
    pHead->next = head;
```

```
    list_node* prevNode = pHead;
    for(int i = 0;i < L-1;i++)
    {
        prevNode = prevNode->next;
    }
    list_node* cur = prevNode->next;
    for(int i = 0;i < R-L;i++)
    {
        list_node* nextNode = cur->next;
        cur->next = nextNode->next;
        nextNode->next = prevNode->next;
        prevNode->next = nextNode;
    }
    list_node* list = pHead->next;
    free(pHead);
    return list;
}
```

day28-22612 奇数位上都是奇数或者偶数位上都是偶数【作废】

https://www.nowcoder.com/questionTerminal/b89b14a3b5a94e438b518311c5156366

【题目解析】： 一个数组的奇数位存奇数，偶数位存偶数

【解题思路】： 在偶数位上寻找非偶数，在奇数位上寻找非奇数，两个数字进行位置互换即可

【示例代码】：

需要注意的是牛客网上答案检测并不严谨，最终有可能出现结果没问题但只是因为与给定答案的数字顺序不同而导致不通过....

```
class Solution {
public:
    void oddInOddEvenInEven(std::vector<int>& arr, int len) {
        long i = 0, j = 1;
        while(i < len && j < len){
            if((arr[i] % 2) == 0) { i += 2; continue; }//偶数位上寻找非偶数
            if((arr[j] % 2) != 0) { j += 2; continue; }//奇数位上寻找非奇数
            swap(arr[i], arr[j]);
        }
    }
};
```

day28-774 猴子分桃

https://www.nowcoder.com/questionTerminal/480d2b484e1f43af8ea8434770811b4a

【题目解析】： 题目很容易理解，桃子每次平均分成五堆但是会多出一个留给老猴子

给定小猴子数量后，求至少要多少桃子才能保证每个小猴子都能分到桃子，及老猴子得到的桃子

【解题思路】： 公式类推

因为每次分5堆都会多出来1个，所以我们借给猴子们4个，以致每次都可以刚好分成5堆 并且，每次给老猴子的桃子都不在我们借出的那4个中，这样最后减掉4就可以得到结果。 假设最初由x个桃子，我们借给猴子4个，则此时有x+4个， 第一个猴子得到（x+4）/5，剩余（x+4）（4/5）个 第二个猴子分完后剩余（x+4）(4/5)^2个 第三个猴子分完后剩余（x+4）(4/5)^3个 依次类推，第n个猴子分完后剩余（x+4）（4/5）^n 要满足最后剩余的为整数，并且x最小，则当 x+4=5^n时，满足要求；此时，x=5^n - 4；老猴子得到的数量为：x+4）*（4/5）^n + n - 4 = 4^n + n - 4 最后的 +n是因为每个小猴子都会多出一个给老猴子，-4是还了借的4个

【示例代码】：

```
#include <iostream>
#include <string>
#include <math.h>

int main()
{
    int n;
    while(std::cin >> n) {
        if (n == 0) break;
        long total = pow(5, n) - 4;
        long left = pow(4, n) + n - 4;
        std::cout << total << " " << left << std::endl;
    }
    return 0;
}
```