# 题目

请编写一个函数，使其可以删除某个链表中给定的（非末尾）节点，你将只被给定要求被删除的节点。

现有一个链表 -- head = [4,5,1,9]，它可以表示为:

**示例 1:**

输入: head = [4,5,1,9], node = 5

输出: [4,1,9]

解释: 给定你链表中值为 5 的第二个节点，那么在调用了你的函数之后，该链表应变为 4 -> 1 -> 9.

**示例 2:**

输入: head = [4,5,1,9], node = 1

输出: [4,5,9]

解释: 给定你链表中值为 1 的第三个节点，那么在调用了你的函数之后，该链表应变为 4 -> 5 -> 9.

**说明:**

链表至少包含两个节点。

链表中所有节点的值都是唯一的。

给定的节点为非末尾节点并且一定是链表中的一个有效节点。

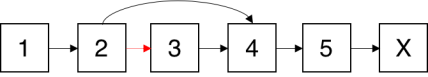
不要从你的函数中返回任何结果。

# 分析

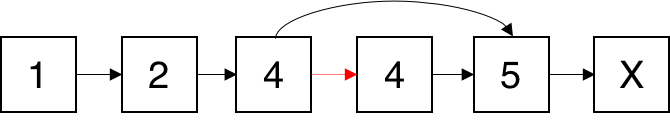
## 方法一：与下一个节点交换

**思路：**

从链表里删除一个节点node的最常见方法是修改之前节点的next指针，使其指向之后的节点。



因为，我们无法访问我们想要删除的节点之前的节点，我们始终不能修改该节点的next指针。相反，我们必须将想要删除的节点的值替换为它后面节点中的值，然后删除它之后的节点。



因为我们知道要删除的节点不是列表的末尾，所以我们可以保证这种方法是可行的。

**代码：**

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* ListNode \*next;

\* ListNode(int x) : val(x), next(NULL) {}

\* };

\*/

class Solution {

public:

void deleteNode(ListNode\* node) {

node->val = node->next->val;

node->next = node->next->next;

}

};

复杂度分析：

时间和空间复杂度都是：O(1)。

可以看出，官方题解的思路是直接将自身替换成了下一个节点

我平时使用C++来做题，但如果简单的将上述代码修改成C++就可能会导致严重的内存泄漏事故

大家都知道Java是有垃圾回收也就是GC的，所以上述代码在Java中是完全没有问题的，因为被替换掉的节点会由GC来释放

而在C++中，每一个被new出来的原生指针(Raw Pinter)都必须使用delete释放，不释放会内存泄漏。

那我们直接写成下面这样：

class Solution {

public:

void deleteNode(ListNode\* node) {

ListNode\* del = node->next;

node->val = node->next->val;

node->next = node->next->next;

delete del;

}

};