题目描述：

给定一个单链表和数值x，划分链表使得所有小于x的节点排在大于等于x的节点之前。

你应该保留两部分内链表节点原有的相对顺序。

输入: list = 1->4->3->2->5->2->null, x = 3

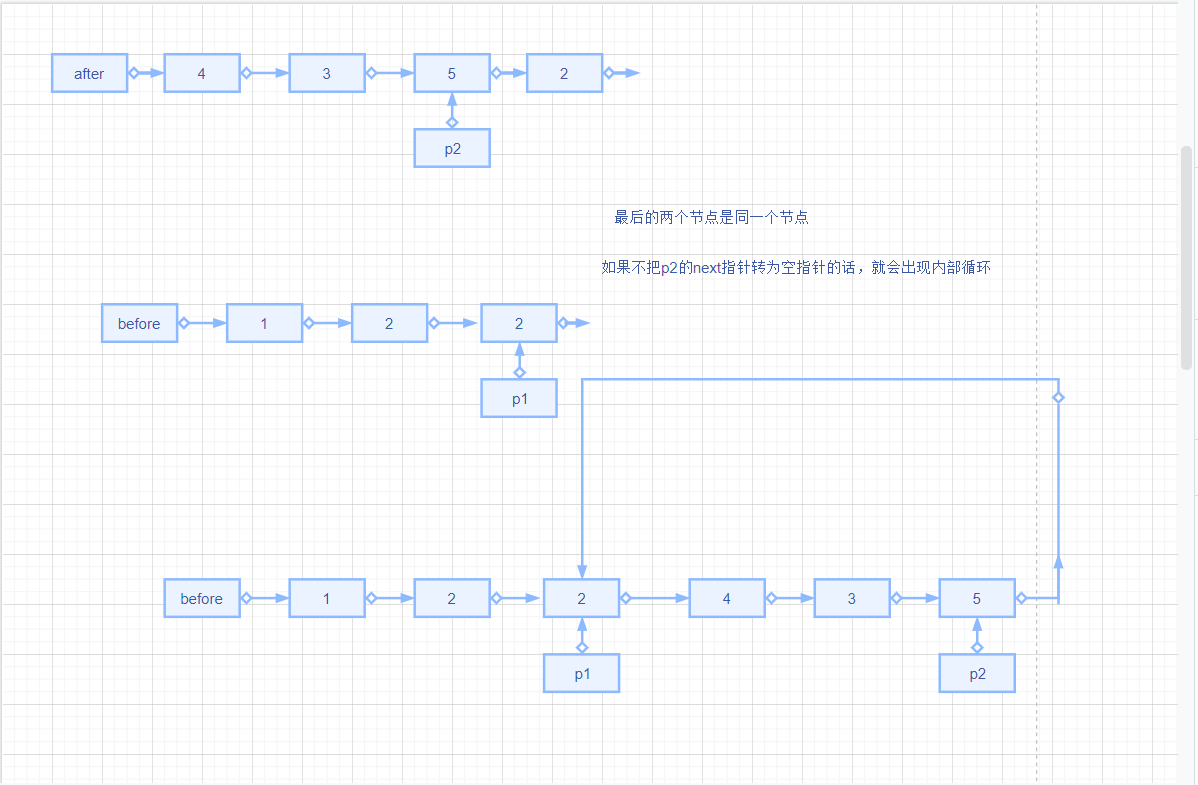
输出: 1->2->2->4->3->5->null

样例解释: 要保持原有的相对顺序。

想法很简单，就是搞两个指针，1根指针指向小于x的，这根指针通过迭代把原链表的小于x的组成一个链表，然后另一根指针指向大的又可以凑一个链表。最后把两个链表一连接，我们就可以解决这道题了。

但是，这里又有一个坑：这个坑会导致链表的局部内循环。如果没有避开这个坑的话，那就会超时报错。很不凑巧的是，我又踩到了这个链表循环的坑。

我用题目给的链表调试了一下，画了下图解：



因此在迭代到最后，如果p2指针的next为空，那当然就不用考虑这个问题了。但是如果next指针不为空的话，就必须要把p2的指针设置为空，然后再进行两表的衔接。

以下是代码：

ListNode\* partition(ListNode\* head, int x) {

// write your code here

ListNode\* before = new ListNode(0);

ListNode\* after = new ListNode(0);

//创建两个指针指向两个新建结点

ListNode\* p1 = before;

ListNode\*p2 = after;

if (!head)

return head;

//遍历链表，<x的值连在before后面，>=x的值连在after后面

while (head)

{

if (head->val < x)

{

p1->next = head;

p1 = p1->next;

head = head->next;

}

else

{

p2->next = head;

p2 = p2->next;

head = head->next;

}

}

//将before和after连起来

p2->next = NULL;

p1->next = after->next;

return before->next;}

截图如下：

