****

**2019级软工暑期算法练习报告**

专业班级： 11J191

学 号： 20191003578

姓 名： 谢凯乐

指导老师： 张剑波

**中国地质大学地理与信息工程学院**

**2020年 8 月**

# 一、链表

## 1、翻转链表（35）

**描述**

翻转一个链表

**样例 1:**

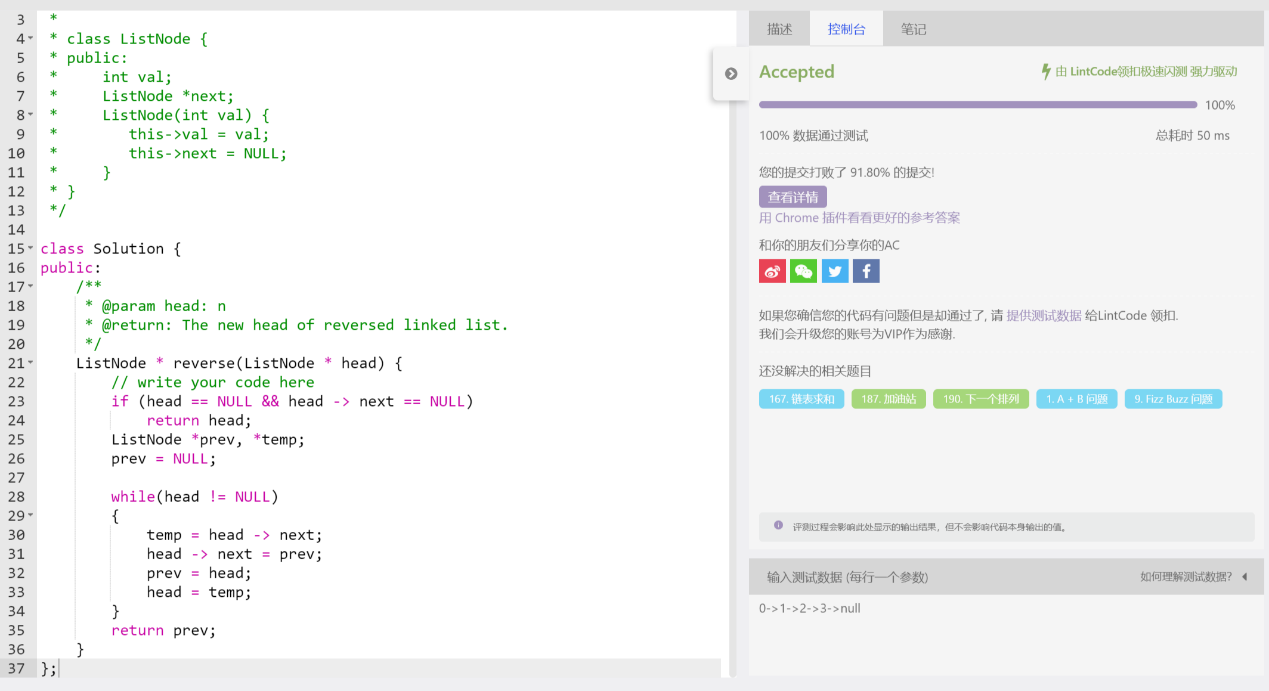
输入: 1->2->3->null

输出: 3->2->1->null

**样例 2:**

输入: 1->2->3->4->null

输出: 4->3->2->1->null



先判断首元结点或第二个节点是否为空，若是，则直接返回head。定义precv和temp两个指针，prev初始化为空指针。用head指针遍历链表每个结点，首先用temp储存head所指结点的后继结点的地址，然后翻转结点，即把prev赋给head所指结点的指针域，然后prev和head都后移一个结点，直到head指向空，此时prev为新链表的头指针，返回prev。

## 2、删除排序链表中的重复元素（112）

**描述**

给定一个排序链表，删除所有重复的元素每个元素只留下一个。

**样例 1:**

输入: null

输出: null

**样例 2:**

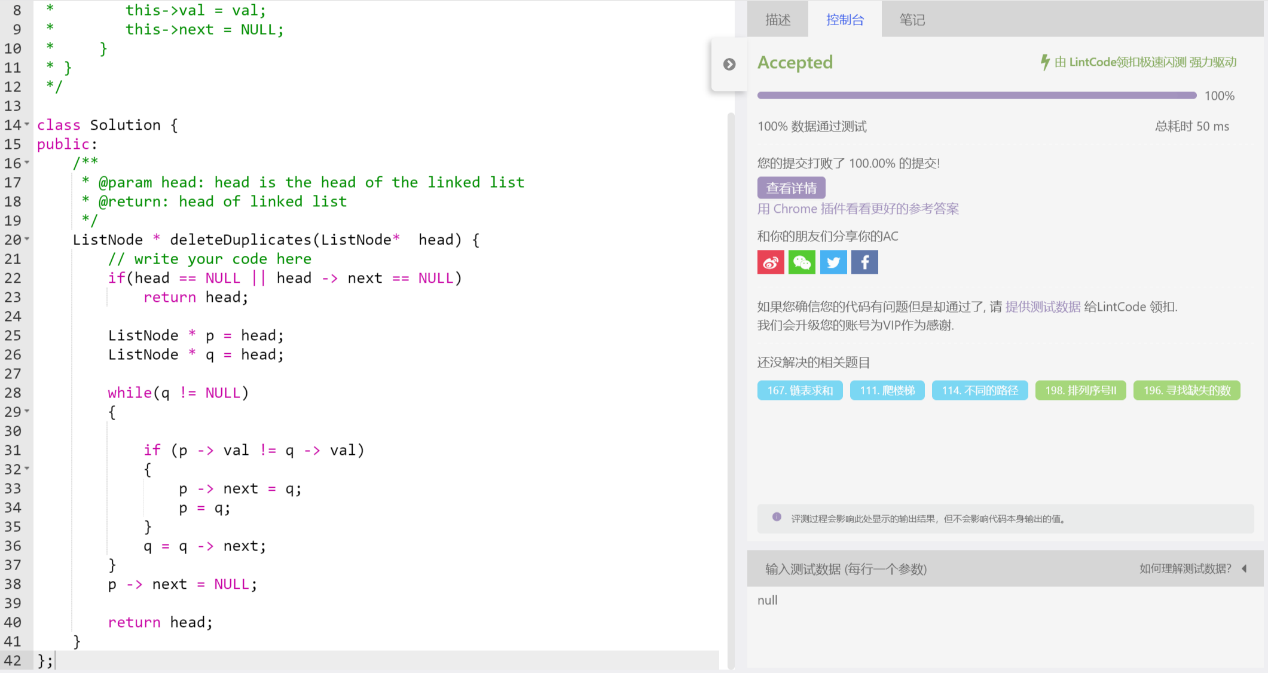
输入: 1->1->2->null

输出: 1->2->null

**样例 3:**

输入: 1->1->2->3->3->null

输出: 1->2->3->null



首先判断首元结点或者第二个结点是否为空，若是，则返回头指针。定义两个指针p和q，都指向首元结点。用q指针遍历整个链表，当q不指向空时，如果q所指结点数值域和p所指结点数值域不相等时，p指向q所指的结点，然后p和q都后移一个结点，如果p和q所指结点数值域相同时，q后移一个结点。循环结束后，p的下一个结点设为空，返回head。

## 3、合并两个排序链表（165）

**描述**

将两个排序链表合并为一个新的排序链表



样例 1**:**

输入: list1 = null, list2 = 0->3->3->null

输出: 0->3->3->null

样例2:

输入: list1 = 1->3->8->11->15->null, list2 = 2->null

输出: 1->2->3->8->11->15->null



首先判断，如果l1为空，返回l2，如果l2为空，返回l1。定义一个头指针head，new一个新结点，定义一个指针curr，把head赋给curr。当l1和l2都不指向空时，比较l1->val和l2->val 的大小，curr指向小的，然后小的和curr都后移一个结点。如果l1指向空，退出循环，curr->next指向l1, 如果l2指向空，退出循环，curr->next指向l2。返回head。

## 4、链表倒数第n个节点（166）

**描述**

找到单链表倒数第n个节点，保证链表中节点的最少数量为n。

**样例**

Example 1:

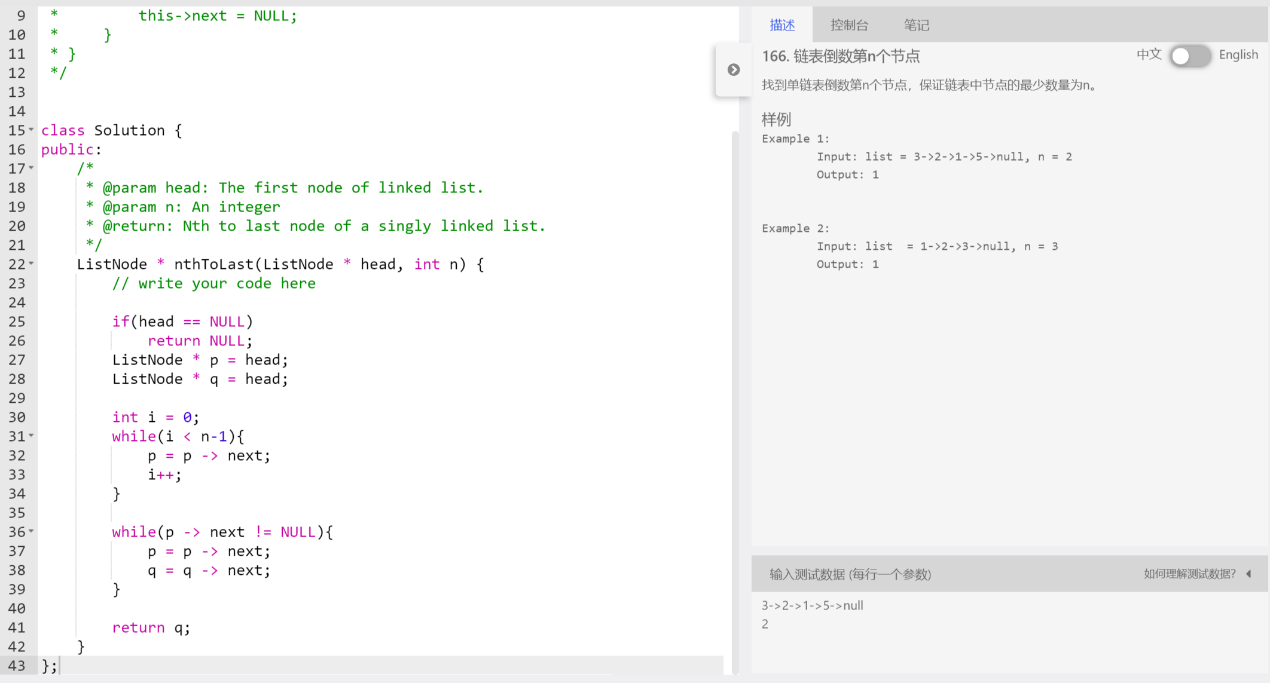
Input: list = 3->2->1->5->null, n = 2

Output: 1

Example 2:

Input: list = 1->2->3->null, n = 3

Output: 1



定义两个指针p和q，把head赋给和q。首先，p走到正数第n个结点。然后，p和q一起走，p走一步，q也走一步。当p走到最后一个结点时，q就走到正数第（链表长度+1-n）个结点，即倒数第n个结点，返回q。

## 5、删除链表中倒数第n个节点（174）

**描述**

给定一个链表，删除链表中倒数第n个节点，返回链表的头节点。

链表中的节点个数大于等于n

**样例**

Example 1:

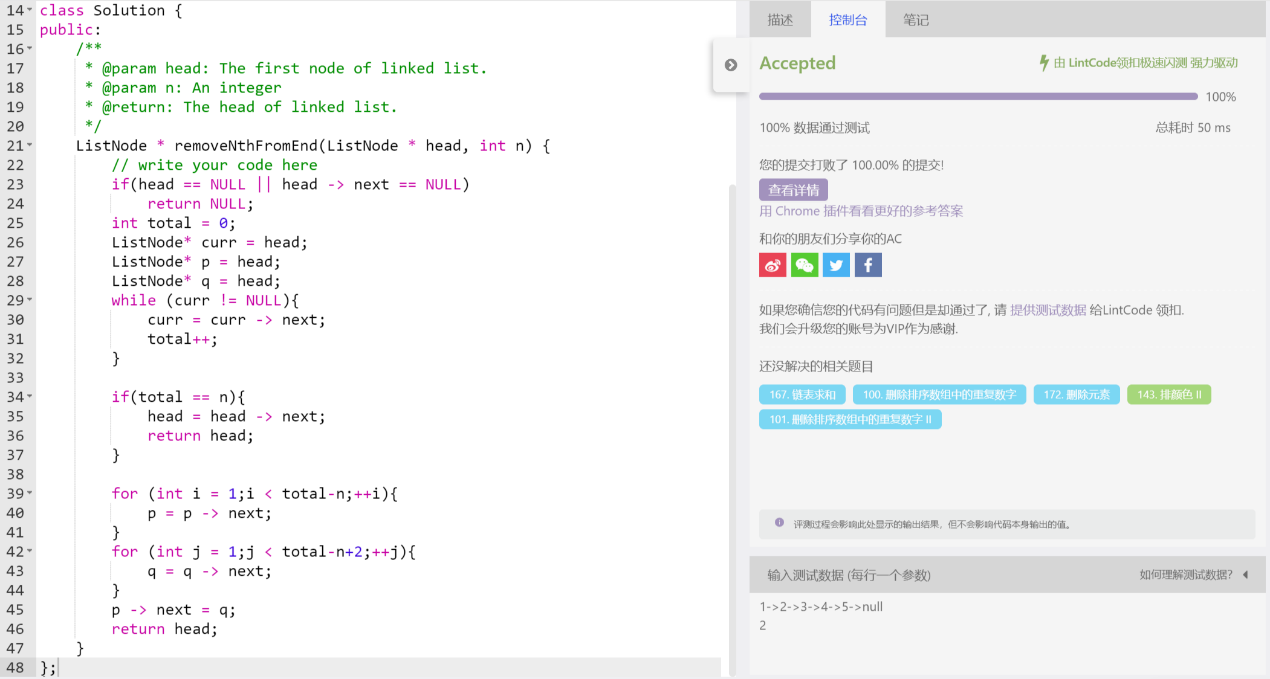
Input: list = 1->2->3->4->5->null， n = 2

Output: 1->2->3->5->null

Example 2:

Input: list = 5->4->3->2->1->null, n = 2

Output: 5->4->3->1->null



首先判断第一个或第二个结点是否为空，若是，返回NULL。然后求出链表长度total。单独考虑删除的结点为首元结点的情况，head后移，返回head。定义两个指针p和q，把head赋给p和q。p走到删除结点的前驱结点，q走到删除结点的后继结点，然后p->next=q, 返回head。

## 6、在排序链表中插入一个节点（219）

**描述**

在链表中插入一个节点。

**样例 1：**

输入：head = 1->4->6->8->null, val = 5

输出：1->4->5->6->8->null

**样例 2：**

输入：head = 1->null, val = 2

输出：1->2->null



插入节点的数值域可能比头结点的小，也可能比尾结点的大，也可能在中间。定义一个指针newnode，new一个空间，把val赋给数值域。当链表为空或者头结点数值域大于等于val时，把头指针赋给newnode->next，返回newnode。定义一个指针curr，赋初值head，从头开始遍历链表，当curr下一个节点大于等于val,或者curr走到了尾结点时，退出遍历。然后把newnode插入curr和curr->next之间，返回head。

## 7、删除链表中的元素（452）

**描述**

删除链表中等于给定值 val 的所有节点。

**样例 1：**

输入：head = 1->2->3->3->4->5->3->null, val = 3

输出：1->2->4->5->null

**样例 2：**

输入：head = 1->1->null, val = 1

输出：null



定义一个新的头指针newhead，申请一个空间，把head赋给头结点的指针域。把newhead赋给指针p。当head不指向空时，如果head->val和val相等，head就后移一个结点，然后p的后继结点改为head所指结点，如果head->val和val不相等，那么head、p均后移一个结点。当head指向空，退出循环，newhead后移，返回newhead。

## 8、在O(1)时间复杂度删除链表节点（372）

**描述**

给定一个单链表中的一个等待被删除的节点(非表头或表尾)。请在在 O(1) 时间复杂度删除该链表节点。

**样例 1：**

输入：

1->2->3->4->null

3

输出：

1->2->4->null

**样例 2：**

输入：

1->3->5->null

3

输出：

1->5->null



把要删除的那个结点发数值域改为其后继结点的数值域，再将要删除结点的指针域改为下下个结点的地址。

## 9、链表划分（96）

**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。

**样例 1:**

输入: list = null, x = 0

输出: null

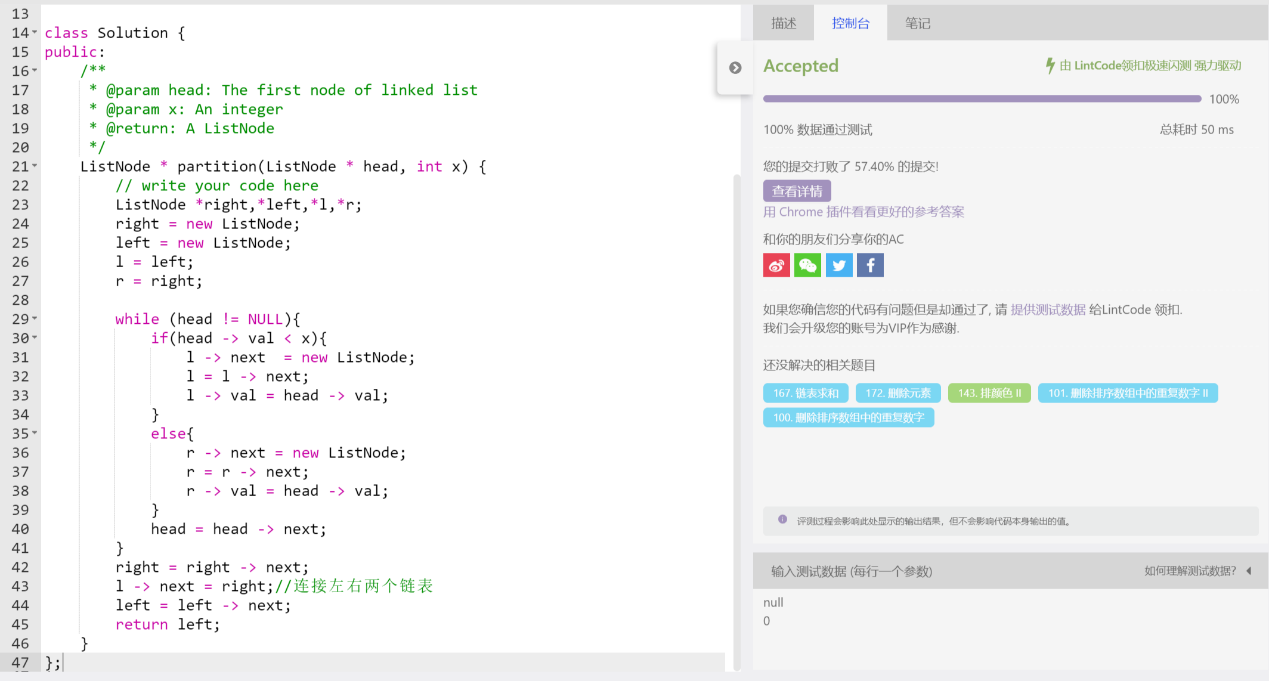
解释: 空链表本身满足要求

**样例 2:**

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

输出: 1->2->2->4->3->5->null

解释: 要保持原有的相对顺序。



用head指针遍历链表的每一个结点，建立两个新链表，一个储存数值域小于x的结点，另一个储存数值域大于等于x的结点，循环结束后，将两个链表连接起来，返回连接后链表的头指针。

## 10、两两交换链表中的节点（451）

**描述**

给一个链表，两两交换其中的节点，然后返回交换后的链表。

**样例 1：**

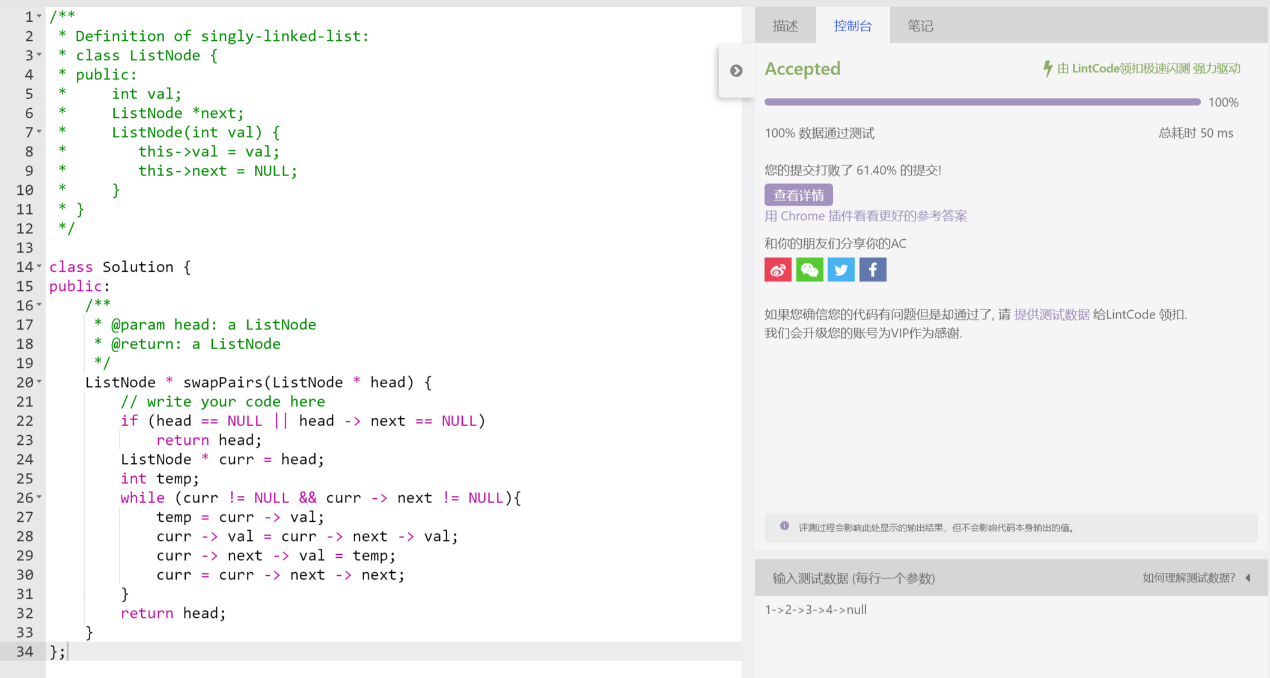
输入：1->2->3->4->null

输出：2->1->4->3->null

**样例 2：**

输入：5->null

输出：5->null



当第一个结点或者只有一个结点时，返回head。定义指针curr，把head赋给curr。定义一个整型变量temp，用来储存currencies所指结点的数值域。然后将curr->val改为后继结点的数值域，再将temp赋给后继结点的数值域，就实现了两两交换。然后curr后移两个结点，直到curr或者curr->next为空，结束循环，返回head。

## 11、链表的中间节点（1609）

**描述**

给定一个带有头结点 head 的非空单链表，返回链表的中间结点。

如果有两个中间结点，则返回第二个中间结点。

The number of nodes in the given list will be between 1 and 100.

**样例 1:**

输入：1->2->3->4->5->null

输出：3->4->5->null

**样例 2:**

输入：1->2->3->4->5->6->null

输出：4->5->6->null



定义指针curr用来遍历链表，首先求出链表长度total，如果total为偶数，curr走到正数第（total/2）个结点，返回curr；如果total为奇数，curr走到正数第（total+1）/2个结点，返回curr。

## 12、相反的顺序存储（822）

**描述**

给出一个链表，并将链表的值以**倒序**存储到数组中。

* 您**不能改变**原始链表的结构。
* ListNode 有两个成员变量：ListNode.val 和 ListNode.next

**样例1**

输入: 1 -> 2 -> 3 -> null

输出: [3,2,1]

**样例2**

输入: 4 -> 2 -> 1 -> null

输出: [1,2,4]



首先求链表长度total，建立数组list[total],用curr从尾到头遍历链表，将结点的数值域按相反的顺序存储到数组中，在建立动态数组List,将数组list[total]的元素挨个添加到动态数组List的末尾，返回List。

## 13、链表求和（167）

**描述**

你有两个用链表代表的整数，其中每个节点包含一个数字。数字存储按照在原来整数中相反的顺序，使得第一个数字位于链表的开头。写出一个函数将两个整数相加，用链表形式返回和。

**样例 1:**

输入: 7->1->6->null, 5->9->2->null

输出: 2->1->9->null

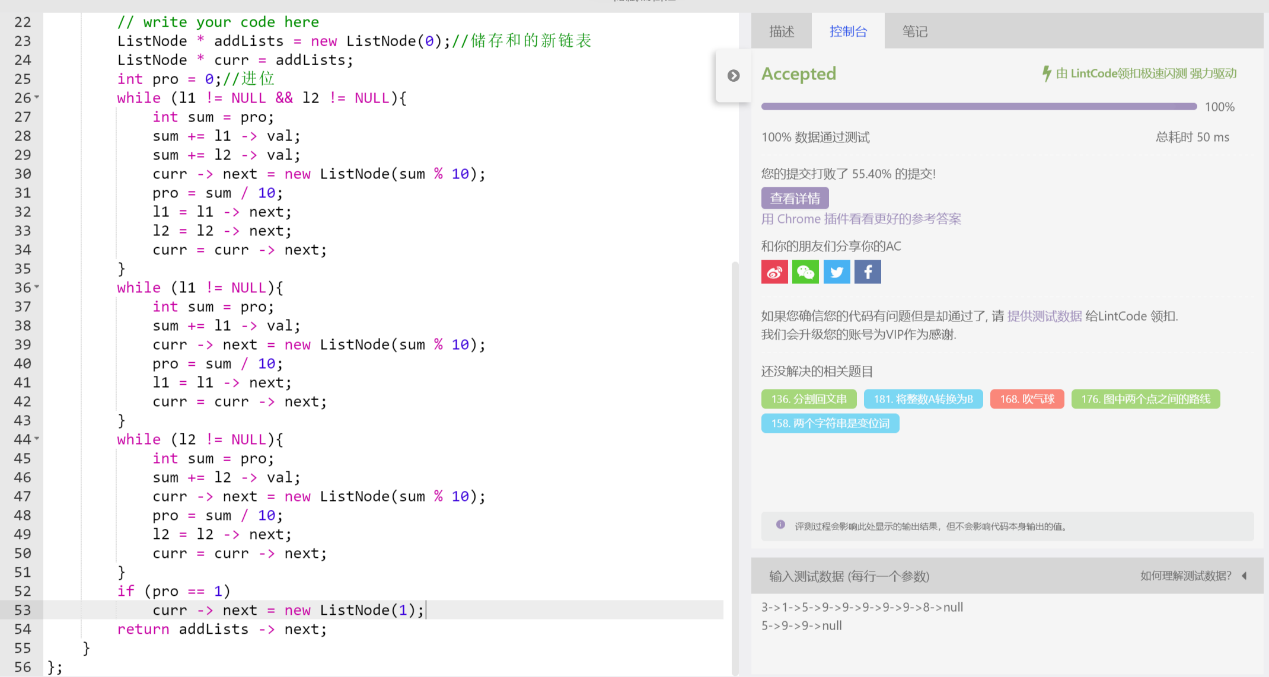
解释: 617 + 295 = 912, 912 转换成链表: 2->1->9->null

**样例 2:**

输入: 3->1->5->null, 5->9->2->null

输出: 8->0->8->null

解释: 513 + 295 = 808, 808 转换成链表: 8->0->8->null



新建一个链表，用来储存和，从个位开始，一位一位的相加，将和除以10得到的余数储存到新链表中，除以10如果等于1，下一位相加时加1，最后返回新链表。

## 14、两数相乘（756）

**描述**

给出两个链表形式表示的数字,写一个函数得到这两个链表相乘乘积。

**样例 1:**

输入：9->4->6->null,8->4->null

输出：79464

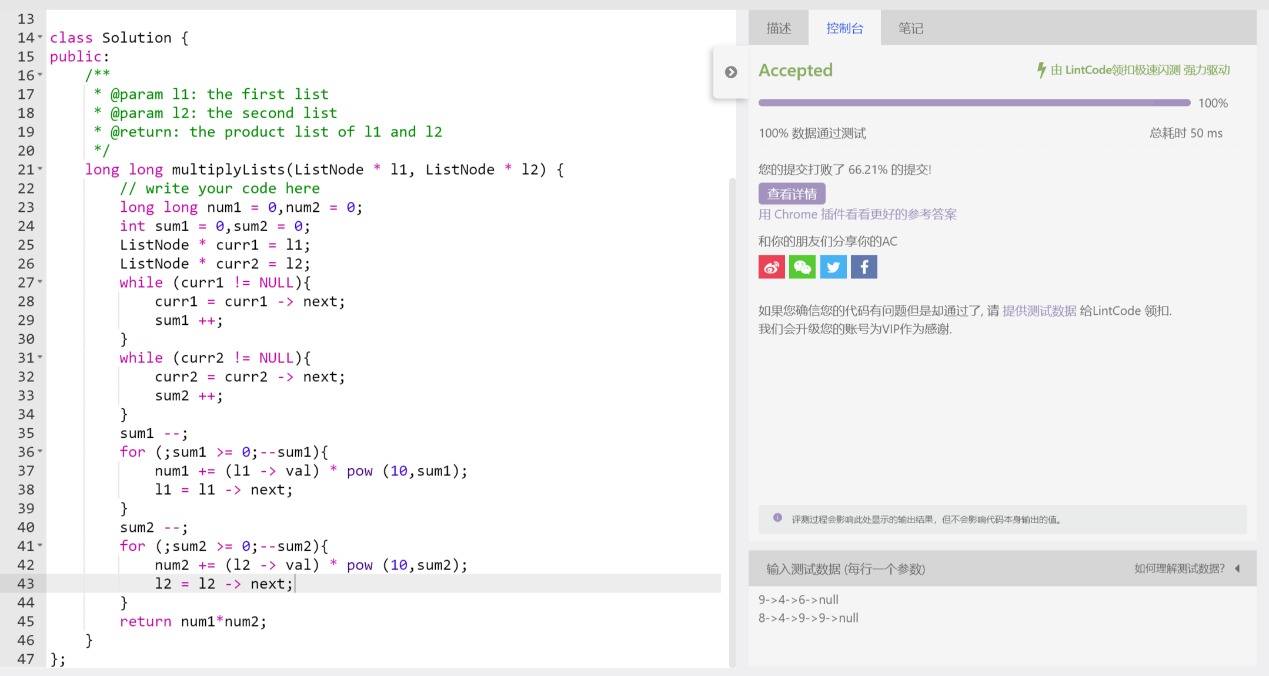
解释：946\*84=79464

**样例 2:**

输入：3->2->1->null,1->2->null

输出：3852

解释：321\*12=3852



把两个链表分别转化为两个整型的数，返回乘积。

## 15、链表插入排序（173）

**描述**

用插入排序对链表排序

**样例**

样例 1:

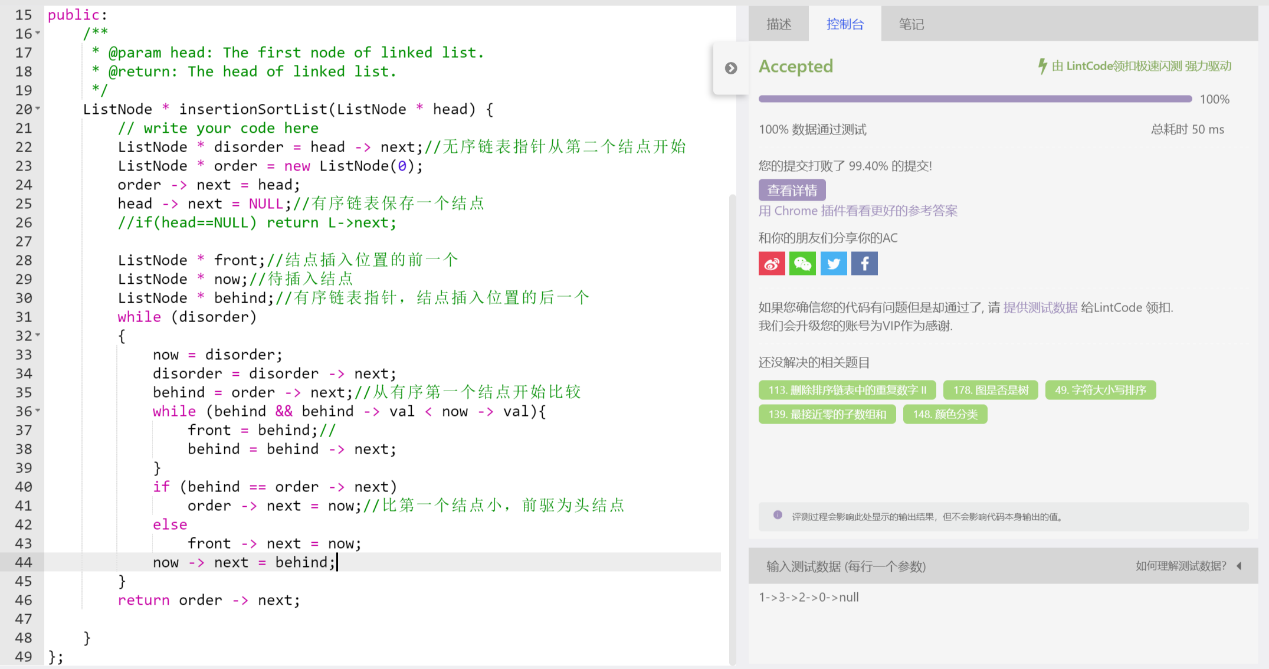
输入: 0->null

输出: 0->null

样例 2:

输入: 1->3->2->0->null

输出: 0->1->2->3->null



建立一个带有头结点的有序链表，还含有原链表的首元结点。建立一个无序链表，头结点为原链表第二个结点。依次将无序链表中的结点插入有序链表

# 二、数组

## 1、合并排序数组Ⅱ（6）

**描述**

合并两个有序升序的整数数组A和B变成一个新的数组。新数组也要有序。

**样例 1:**

输入: A=[1], B=[1]

输出:[1,1]

解释: 返回合并后的数组。

**样例 2:**

输入: A=[1,2,3,4], B=[2,4,5,6]

输出: [1,2,2,3,4,4,5,6]

解释: 返回合并后的数组



建立数组C，依次比较数组A和数组B中的元素，将其中小的元素添加到数组C的末尾，如果数组A或B中的元素遍历完了，就将另一个数组中的元素依次添加到数组C中，返回数组C。

## 2、二分查找（14）

**描述**

给定一个排序的整数数组（升序）和一个要查找的整数target，用O(logn)的时间查找到target第一次出现的下标（从0开始），如果target不存在于数组中，返回-1。

**样例**

样例 1:

输入:[1,4,4,5,7,7,8,9,9,10]，1

输出: 0

解释:

第一次出现在第0个位置。

样例 2:

输入: [1, 2, 3, 3, 4, 5, 10]，3

输出: 2

解释:

第一次出现在第2个位置

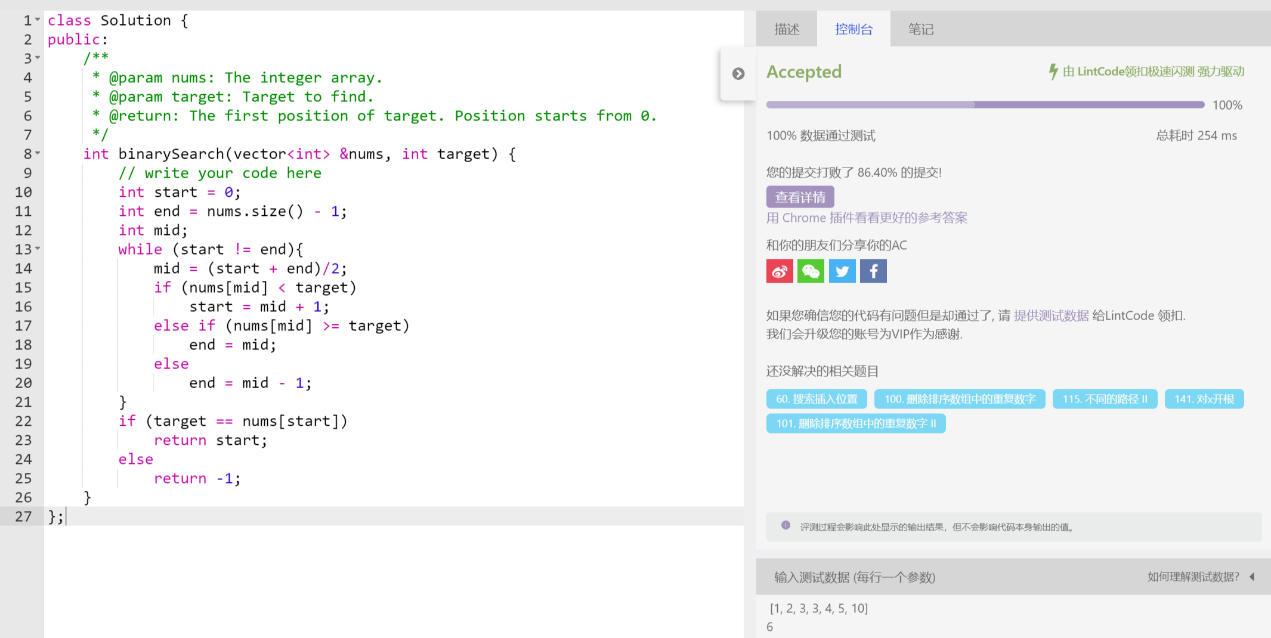
样例 3:

输入: [1, 2, 3, 3, 4, 5, 10]，6

输出: -1

解释:

没有出现过6， 返回-1



当start和end不相等时，若中间位置的值小于target，start=mid+1，若中间位置等于target，end=mid，若中间位置大于target，end=mid-1。当start和end相等时，如果nums[start]==targrt,返回start，否则返回-1。

## 3、恢复旋转排序数组（39）

**描述**

给定一个**旋转**排序数组，在原地恢复其排序。（升序）

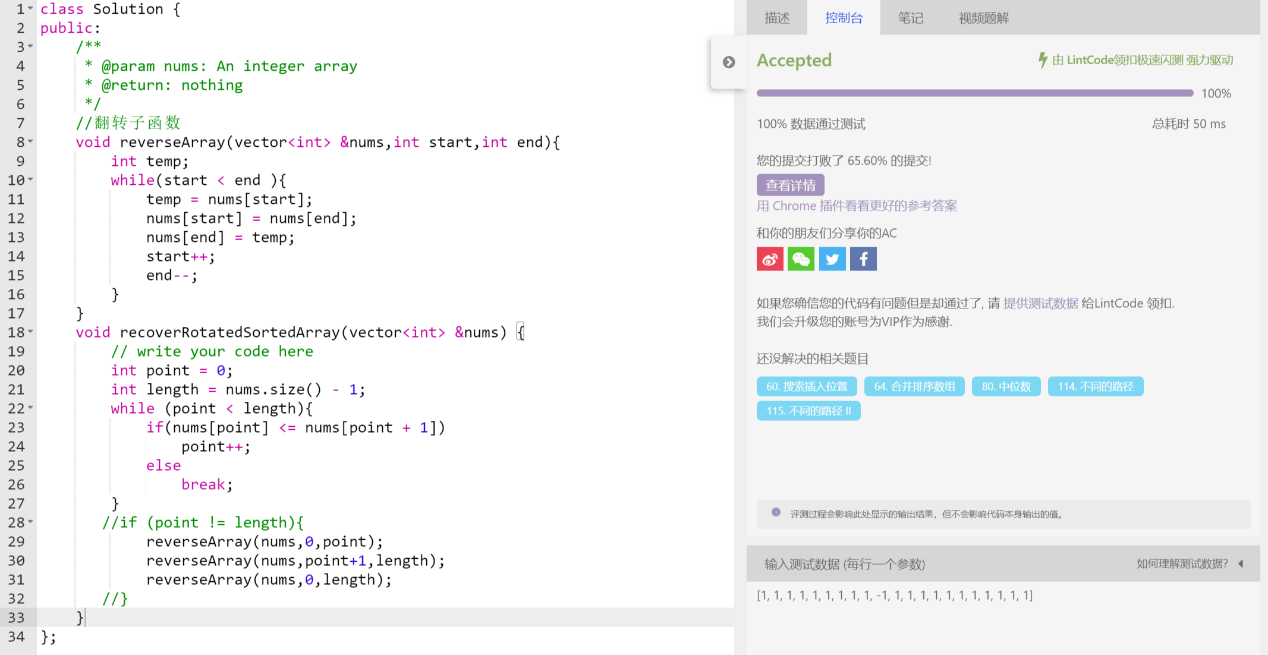
**说明**

什么是旋转数组？

* 比如，原始数组为[1,2,3,4], 则其旋转数组可以是[1,2,3,4], [2,3,4,1], [3,4,1,2], [4,1,2,3]

**样例**

**Example1:**  
[4, 5, 1, 2, 3] -> [1, 2, 3, 4, 5]  
**Example2:**  
[6,8,9,1,2] -> [1,2,6,8,9]



先找到突然减小的断点，然后进行三次翻转，断点前包含断点翻转一次，断点后翻转一次，全体翻转一次。如果没有断点，可以不进行翻转。

## 4、两数之和（56）

**描述**

给一个整数数组，找到两个数使得他们的和等于一个给定的数 *target*。

你需要实现的函数twoSum需要返回这两个数的下标, 并且第一个下标小于第二个下标。注意这里下标的范围是 0 到 *n-1*。

你可以假设只有一组答案。

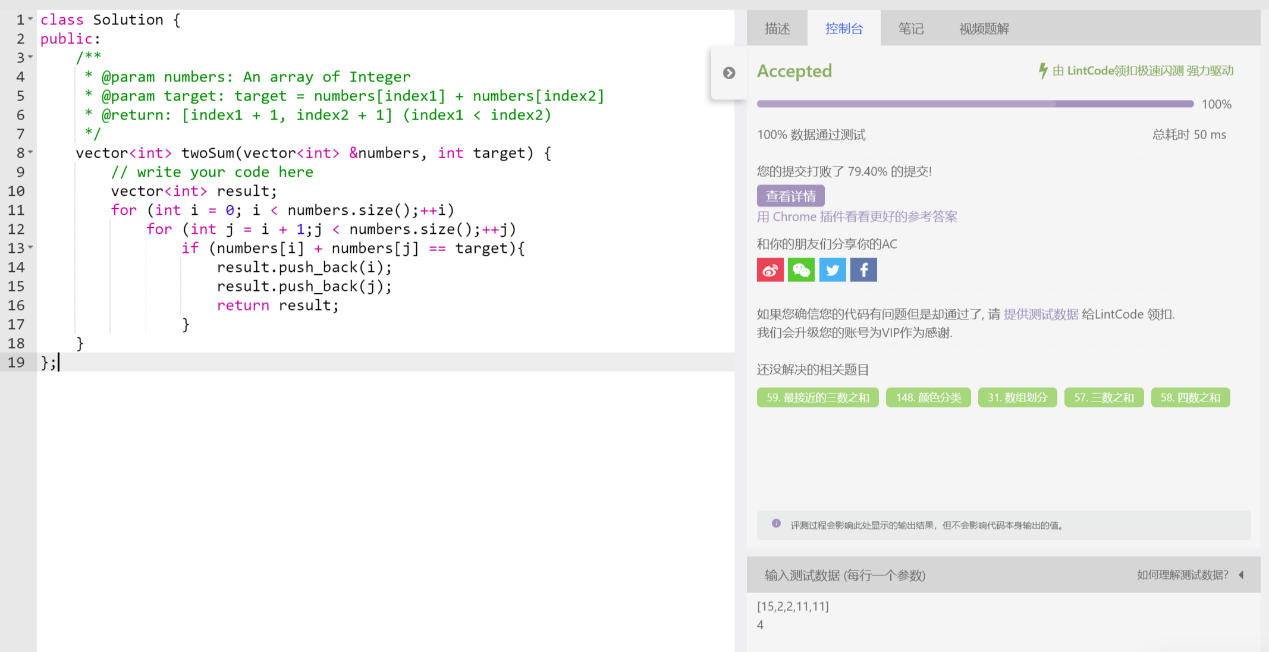
**样例**

Example1:

给出 numbers = [2, 7, 11, 15], target = 9, 返回 [0, 1].

Example2:

给出 numbers = [15, 2, 7, 11], target = 9, 返回 [1, 2].



建立一个新的数组result储存下标，两个for循环遍历数组元素，第二个遍历的起点在第一个之后，找到两个元素和等于target，将下标push\_back到result中。

## 5、搜索插入位置（60）

**描述**

给定一个排序数组和一个目标值，如果在数组中找到目标值则返回索引。如果没有，返回到它将会被按顺序插入的位置。

**样例**

**[1,3,5,6]**，5 → 2

**[1,3,5,6]**，2 → 1

**[1,3,5,6]**，7 → 4

**[1,3,5,6]**，0 → 0



当数组为空时，直接返回0；遍历数组，遇到比target小的，直接遍历到下一个元素，遇到大于等于target的，直接返回索引值，遍历完也没有大于等于target的，返回数组长度。

## 6、合并排序数组（64）

**描述**

合并两个排序的整数数组A和B变成一个新的数组。

你可以假设A具有足够的空间（A数组的大小大于或等于m+n）去添加B中的元素。

**样例 1:**

输入：[1, 2, 3] 3 [4,5] 2

输出：[1,2,3,4,5]

解释:

经过合并新的数组为[1,2,3,4,5]

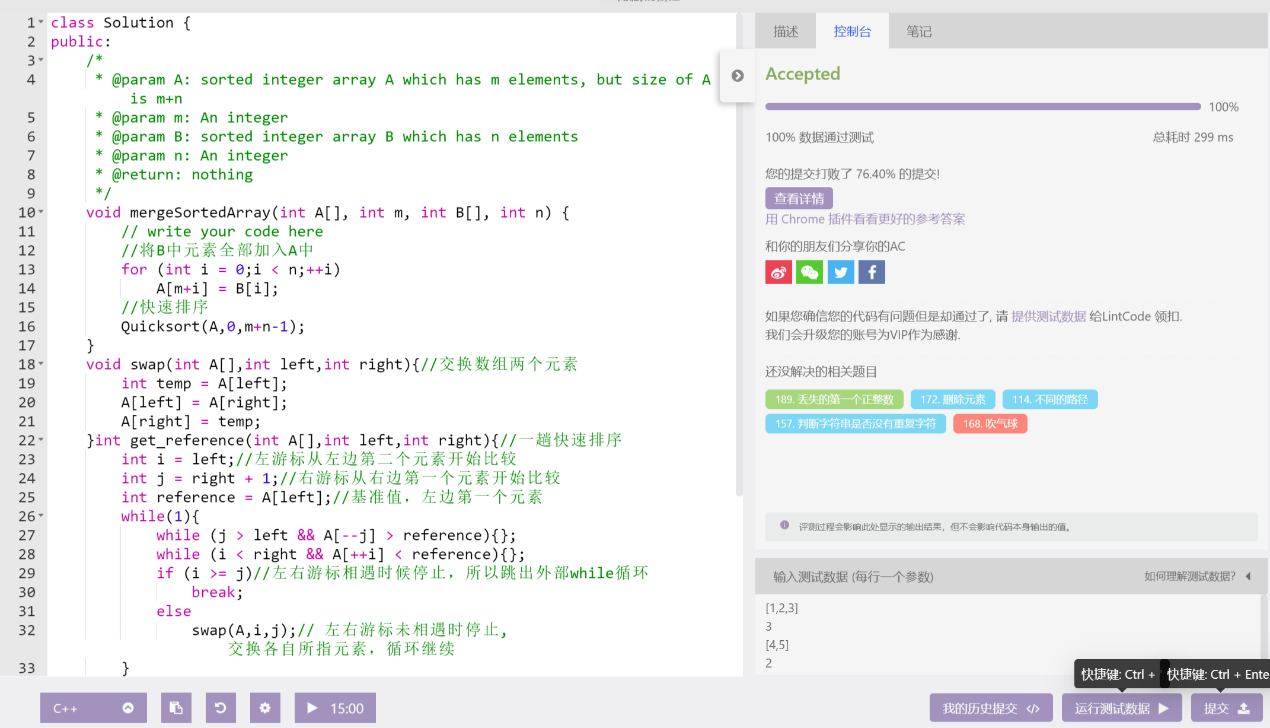
**样例 2:**

输入：[1,2,5] 3 [3,4] 2

输出：[1,2,3,4,5]

解释：

经过合并新的数组为[1,2,3,4,5]



先把B中的元素全部添加到A中，然后再进行快速排序。

## 7、中位数（80）

**描述**

给定一个未排序的整数数组，找到其中位数。

中位数是排序后数组的中间值，如果数组的个数是偶数个，则返回排序后数组的第N/2个数。

数组大小不超过10000

**样例 1:**

输入：[4, 5, 1, 2, 3]

输出：3

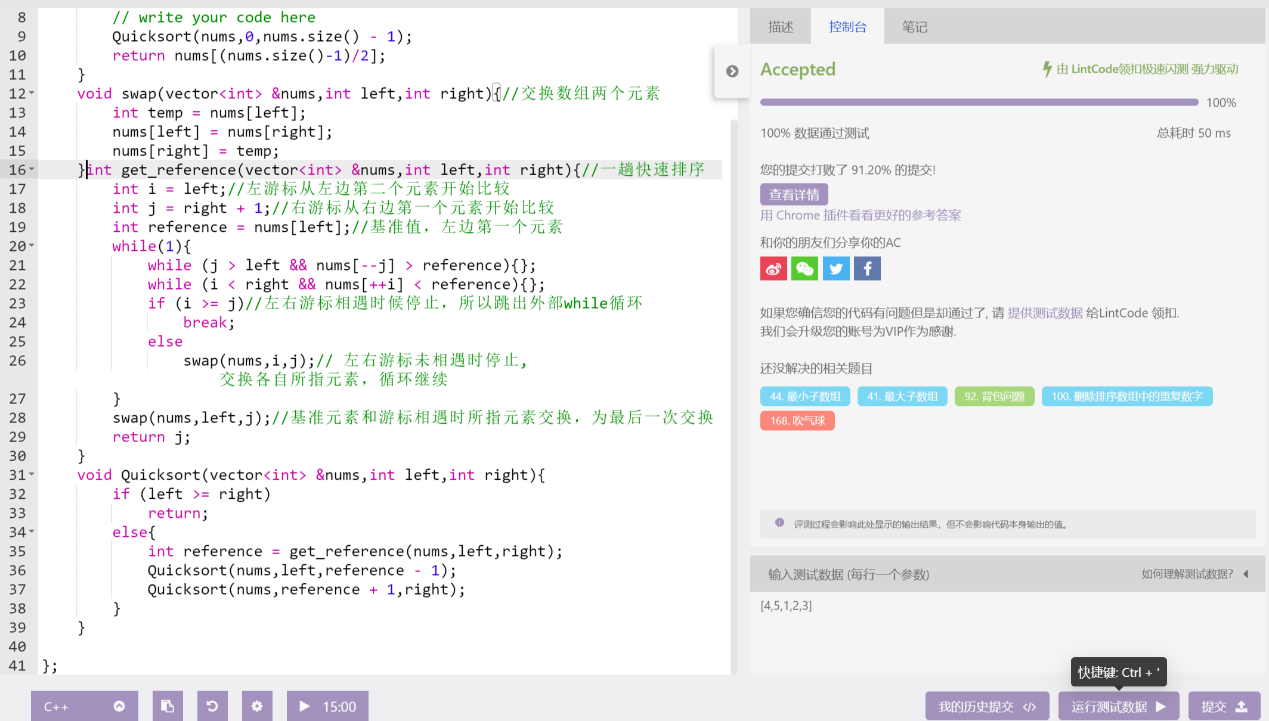
解释：经过排序，得到数组[1,2,3,4,5]，中间数字为3

**样例 2:**

输入：[7, 9, 4, 5]

输出：5

解释：经过排序，得到数组[4,5,7,9]，第二个(4/2)数字为5



使用快速排序对数组进行排序，得到升序排序的数组，然后返回nums[(nums.size()-1)/2]。