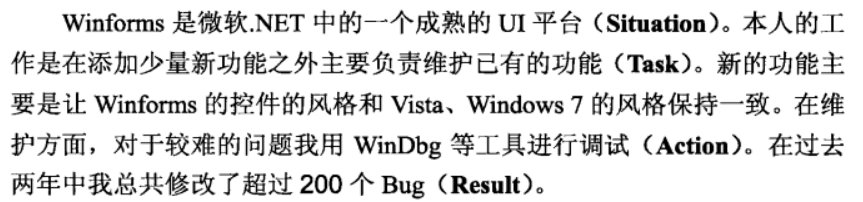
## 注意事项：

2、对公司近况、项目情况有所了解，准备好合适的问题问面试官。

3、项目经验：简短的项目背景（项目的规模，功能，目标用户等）、**完成的任务**（负责：项目的总体框架设计、核心算法、团队合作）、为了完成任务做了哪些工作，怎么做的（基于什么工具什么平台下应用了哪些技术）、**自己的贡献**（完成了多少功能，性能优化提高的百分比）。



项目中碰到的最大的问题是什么，怎么解决的；学到了什么；解决冲突。

4、参与：加入某一个开发团队写了几行代码；

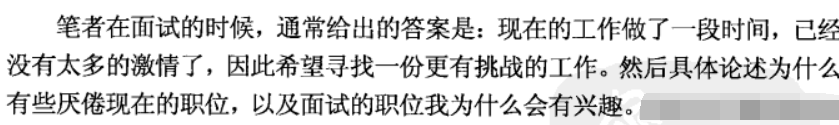
负责：项目的总体框架设计、核心算法、团队合作

了解：只是上过课或者看过书，没有做过实际的项目。【建议少用】

熟悉：在实际项目中使用已经有较长时间，通过查阅相关的文档可以独立解决大部分问题。

精通：轻松回答这个领域里的绝大数问题。

5、为什么跳槽：



1、**编码前将分析过程，自己的思路，明白要做的是什么，怎么做，可以采用举例子、画图**；

2、位于运算（&1）比求约判断奇偶效率高；

3、(n-1) & n 得到的结果相当于把n的二进制表示中的最右边一个1变成0。

4、float、double不能直接==比较大小

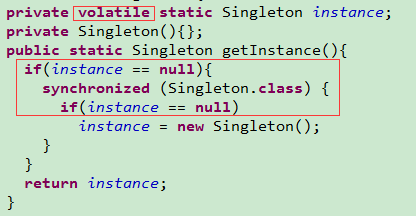
5、大数据可以使用字符串模拟，以免溢出，打印时去掉前面的0.

6、链表 随时考虑null 和只有一个节点情况。

7、扩展性的考虑

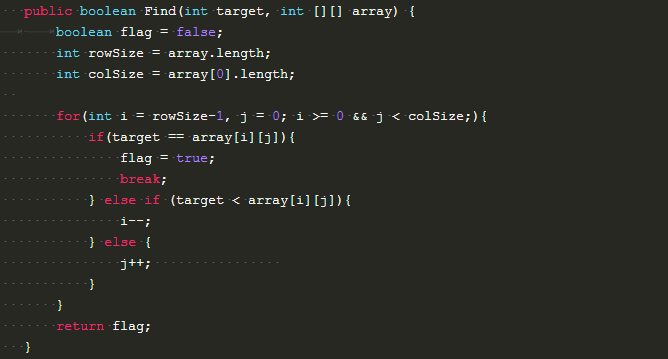
## 题目1:实现Singleton模式

题目描述：设计一个类，我们只能生成该类的一个实例。



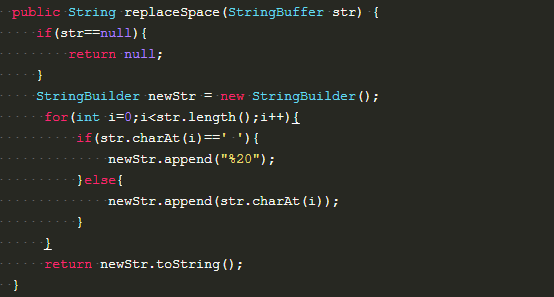
## 题目2:二维数组中的查找

题目描述：在一个二维数组中，每一行都按照从左到右递增的顺序排序，每一列都按照从上到下递增的顺序排序。请完成一个函数，输入这样的一个二维数组和一个整数，判断数组中是否含有该整数。



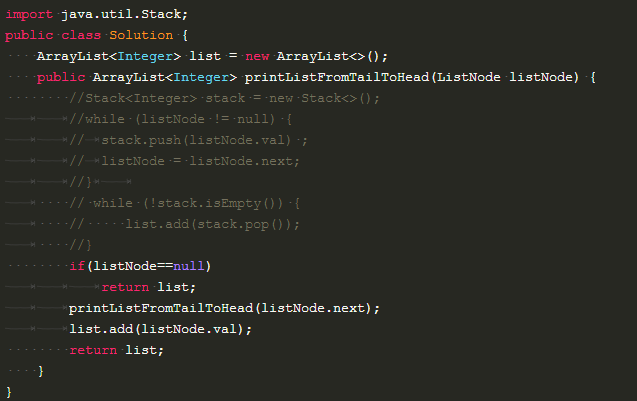
## 题目3：替换空格

题目描述：请实现一个函数，将一个字符串中的空格替换成“%20”。例如，当字符串为We Are Happy.则经过替换之后的字符串为We%20Are%20Happy。



## 题目4：从尾到头打印链表

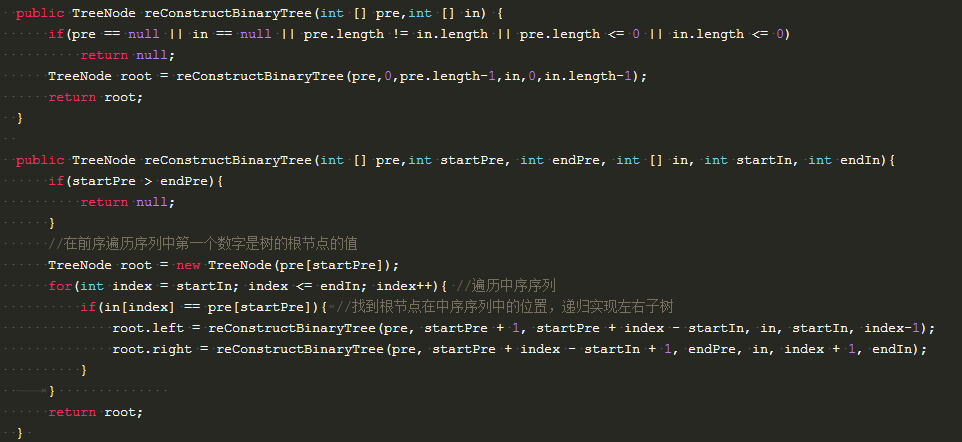
题目描述：输入一个链表，从尾到头打印链表每个节点的值。【栈更好】



## 题目5：重建二叉树

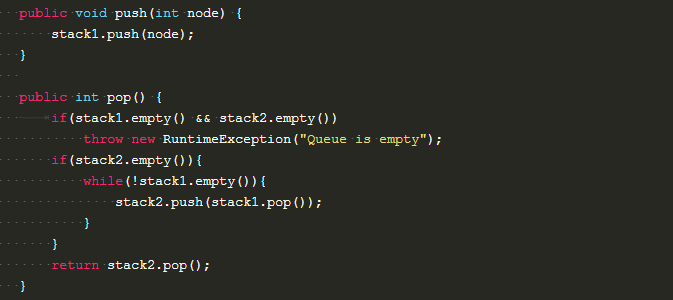
题目描述：输入某二叉树的前序遍历和中序遍历的结果，请重建出该二叉树。假设输入的前序遍历和中序遍历的结果中都不含重复的数字。例如输入前序遍历序列{1,2,4,7,3,5,6,8}和中序遍历序列{4,7,2,1,5,3,8,6}，则重建二叉树并返回。

注意：在前序遍历序列中，第一个数字总是树的根节点的值。中序遍历中，根节点的值中间。左子树的节点的值位于根节点的值的左边，而右子树的节点的值位于根节点的值的右边。



## 题目6：用两个栈实现队列

题目描述：用两个栈来实现一个队列，完成队列的Push和Pop操作。 队列中的元素为int类型。



## 题目7：旋转数组的最小数字

注意：如果要求在排序的数组中查找一个数字或者统计某个数字出现的次数，可以尝试用**二分查找**算法。排序：插入排序、冒泡排序、归并排序、**快速排序**。能够从额外的空间消耗、平均时间复杂度和最差时间复杂度等方面进行比较。

快速排序：在数组中选择一个数字，接下来把数组中的数字分为两部分，比选择的数字小的数字移到数组的左边，比选择的数字大的数字移到数组的右边。

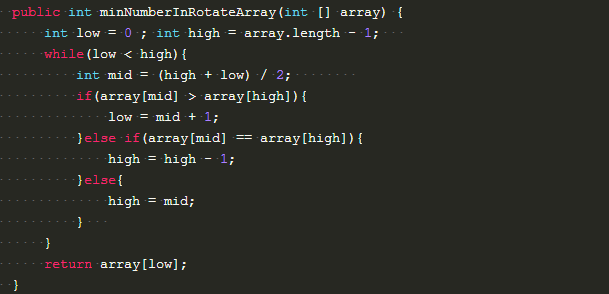
**解析二分法算法： mid = (low + high) / 2;**

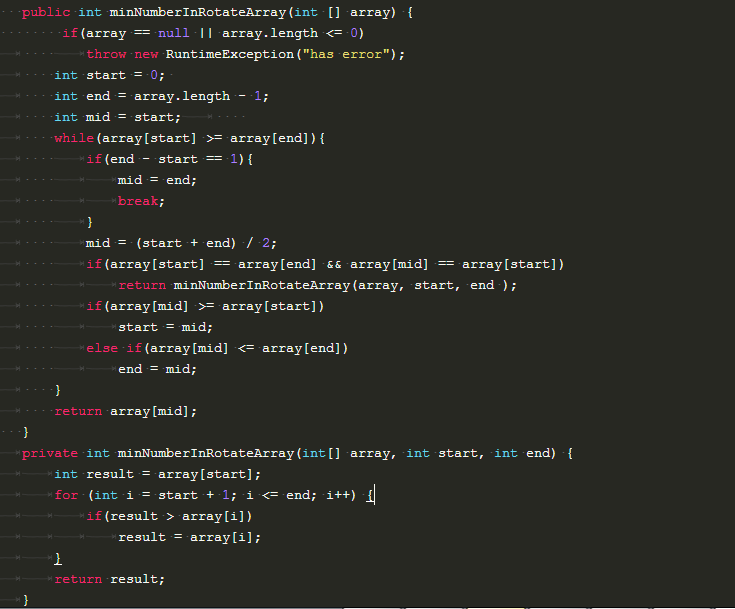
**①array[low]<array[high] ，结果为array[low]**

**②只有两个数 ，结果为array[high]**

**③array[mid] >= array[low] 前半部递增，结果处于后半段，即low = mid;**

**④array[mid] <= array[high] 后半部递增，结果处于前半段，即high= mid;**



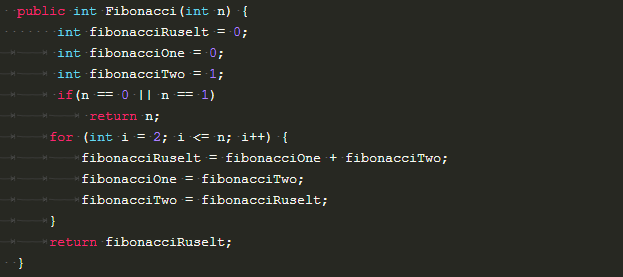


注意：基于递归实现的代码比基于循环实现的代码要简介很多。但是递归需要在内存占中分配空间以保存参数、返回地址及临时变量，往往**效率不如循环**。递归还可能引起更严重的问题：**调用栈溢出**。

## 题目8：斐波那契数列

要求输入一个整数n，请你输出斐波那契数列的第n项。 F(n) = n (n=0 || n =1);

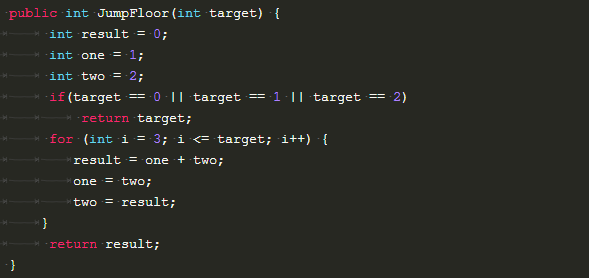
F(n) = F(n - 1) + F(n - 2)



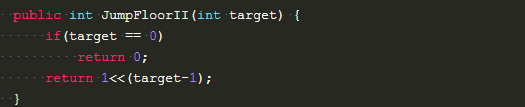
## 举一反三：跳台阶

1、一只青蛙一次可以跳上一级台阶，也可以跳上2级，请问该青蛙跳上一个n级的台阶总共有多少种跳法。

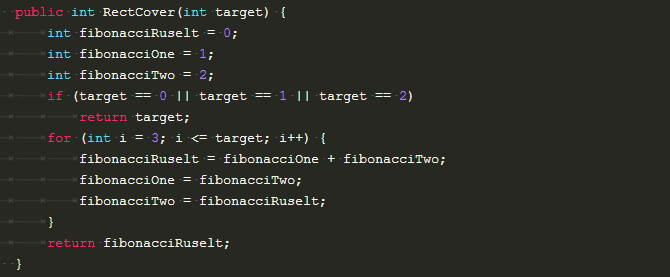
解答：如果只有1级台阶，那显然只有一种跳法，如果有2级台阶，就有两种跳的方法。如果是n级台阶，记住f(n)，当第一次跳的时候，就有两种跳法：①第一次只跳1级，此时跳法有f(n-1)种；②如果第一次跳2级，此时跳法有f(n-2)种。则F(n) = F(n - 1) + F(n - 2)。



2、如果可以一次跳1级，可以一次跳2级，… …，可以一次跳n级，则F(n) = 2 的n-1次方。

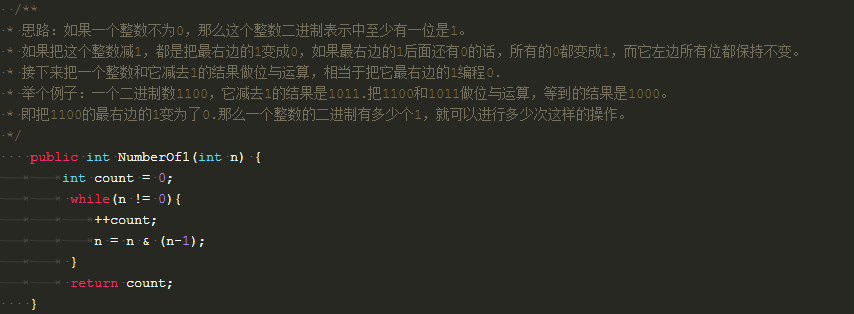


3、用2\*1的矩形去覆盖8\*2的矩形方法。 F(n) = F(n-1)+F(n-2)



## 题目9：二进制中1的个数

题目描述：输入一个整数，输出该数二进制表示中1的个数。其中负数用补码表示。



## 举一反三：是不是2的整数次方

1、用一条语句判断一个整数是不是2的整数次方。

解答思路：一个整数如果是2的整数次方，那么它的二进制表示中有且只有一位是1，二其他所有位都是0.

答案： return n & (n-1) == 0.

2、输入两个整数m和n，计算需要修改m的二进制表示中的多少位才能得到n。

解答思路：第一步求这两个数的异或，第二步统计异或结果中1的位数。

## 题目10：数值的整数次方

实现函数 double Power(double BASE, int exponent),即给定一个double类型的浮点数base和int类型的整数exponent。求base的exponent次方。

思路：exponent=0 或者 <0

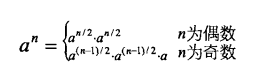
exponent<0，求导数，可能出现对0求导数

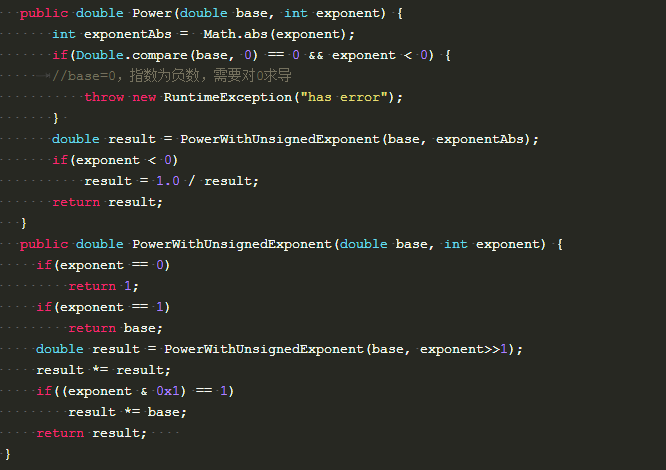
base=0，exponent<0 ？？

注意：0的0次方 没有意义

当判断base是不是等于0时，不能直接写base==0【float和double不能直接用等号判断两个小数是否相等】

使用右移元素那代替除以2，用位于运算代替求余来判断奇偶.

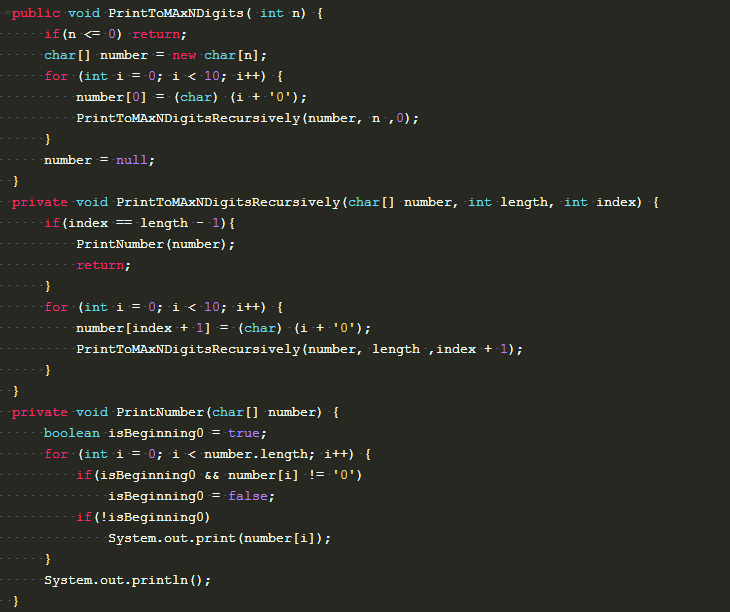




## 题目11：打印1到最大的n位数

\* 打印1到最大的n位数

\* 思路：考虑越界，使用字符串或者数组表达大数



## 举一反三：两个整数相加？？？

题目描述：任意两个整数相加

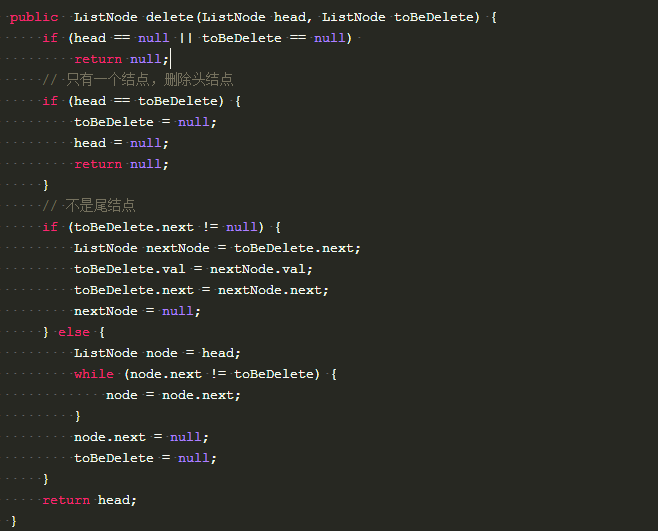
## 题目12：在O(1)时间删除链表节点

题目描述：给定单向链表的头指针和一个节点指针，定义一个函数在O(1)时建删除该结点。

思路：如果需要删除结点i ，先把i的下一个结点j的内容复制到i，然后把i的下一个结点指向j的下一个结点，删除j。如果要删除的结点位于链表的尾部，那么他就没有下一个结点，仍然需要从链表的头结点开始，顺序遍历得到该结点的前序结点，并完成删除。如果，链表中只有一个结点，删除结点之后，需要把链表的头结点设置为null。

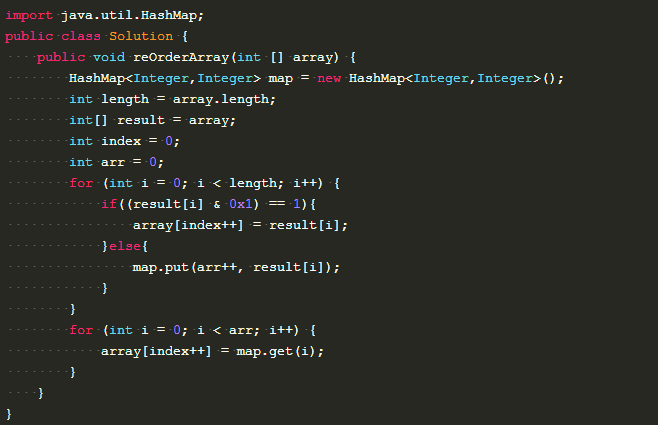
时间复杂度：[(n-1) \* O(1) + O(n)]/n = O(1)

前提是：假设删除的结点存在于链表中



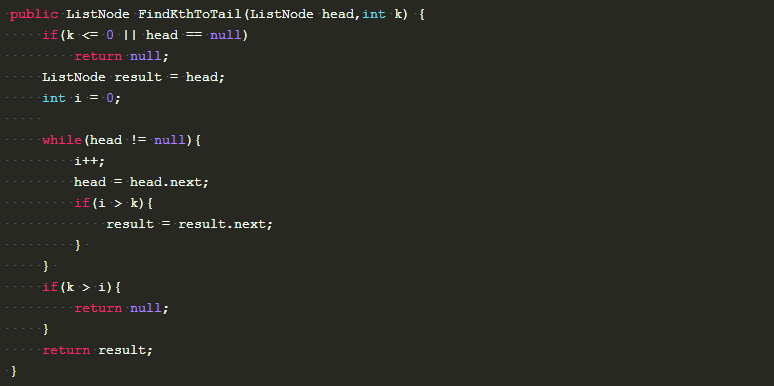
## 题目13：调整数组顺序使奇数位于偶数前面

题目描述：输入一个整数数组，实现一个函数来调整该数组中数字的顺序，使得所有的奇数位于数组的前半部分，所有的偶数位于位于数组的后半部分，并保证奇数和奇数，偶数和偶数之间的相对位置不变。



## 题目14：链表中倒数第k个结点

题目描述：输入一个链表，输出该链表中倒数第k个结点。

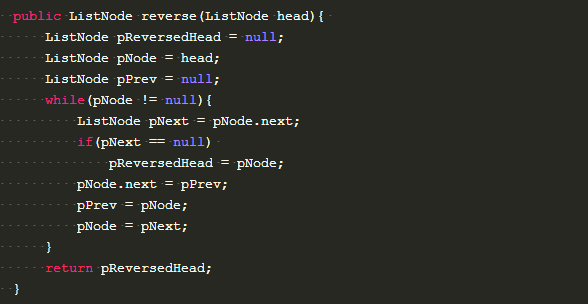


## 举一反三：使用两个指针

1、求链表中间节点。可以定义两个指针，同时从链表的头结点出发，一个指针一次走一步，另一个指针一次走两步。当走的快的指针走到链表的末尾时，走得慢的指针正好在链表的中间。

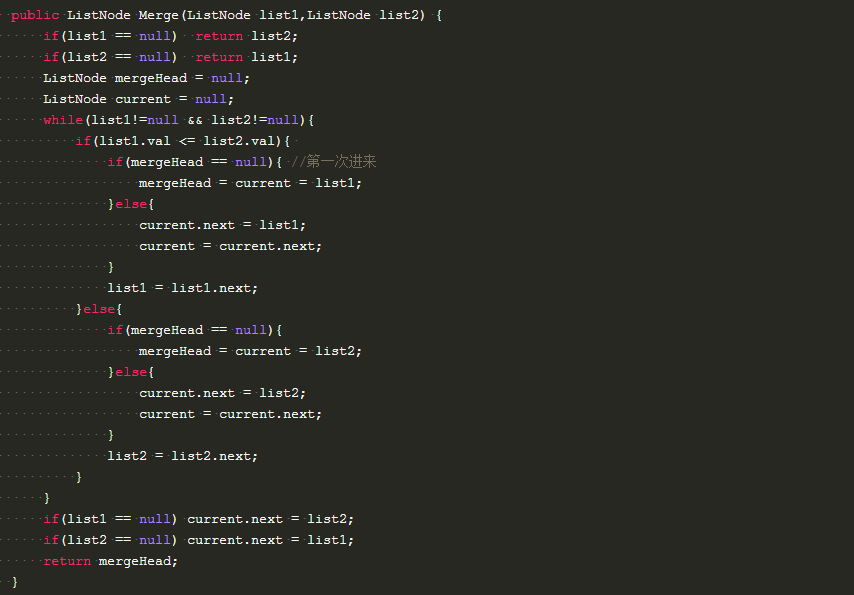
2、判断一个单向链表是否形成了环形结构。定义两个节点从头结点出发，一个指针一次走一步，另一个一次走两步，如果走得快的指针追上了走的慢的指针，那么链表就是环形链表；如果走的快的指针走到了链表的末尾都没有追上第一个指针，那么链表就不是环形链表。

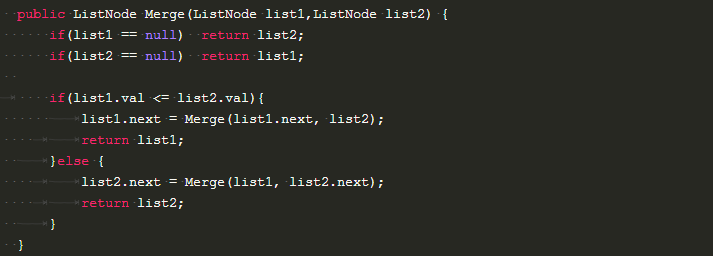
## 题目15：反转链表



## 题目16：合并两个排序的链表

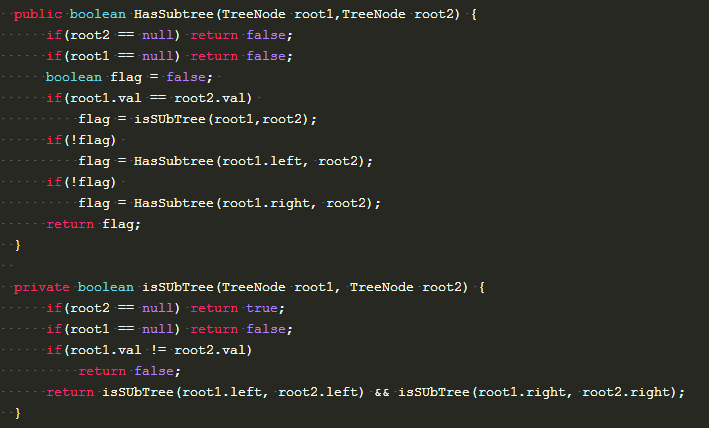
题目描述：输入两个单调递增的链表，输出两个链表合成后的链表，当然我们需要合成后的链表满足单调不减规则。



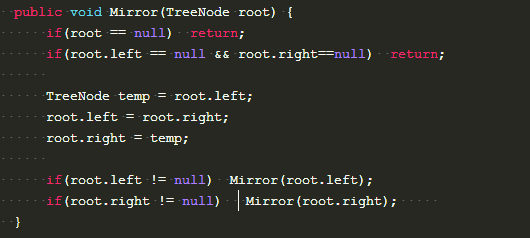


## 题目17：树的子结构

题目描述：输入两棵二叉树A，B，判断B是不是A的子结构。（ps：我们约定空树不是任意一个树的子结构）。



## 题目18：二叉树的镜像



## 题目19

## 题目20

## 题目21

## 题目22

## 题目23

## 题目24

## 题目25

## 题目26

## 题目27

## 题目28

## 题目29

## 题目30

## 题目31

## 题目32

## 题目33

## 题目34

## 题目35

## 题目36

## 题目37

## 题目38

## 题目39

## 题目40

## 题目41

## 题目42

## 题目43

## 题目44

## 题目45

## 题目46

## 题目47

## 题目48

## 题目49

## 题目50

## 题目51

## 题目52

## 题目53

## 题目54

## 题目55

## 题目56

## 题目57

## 题目58

## 题目59

## 题目60

## 题目61

## 题目62

## 题目63

## 题目64

## 题目65

## 题目66

## 题目67

## 题目68