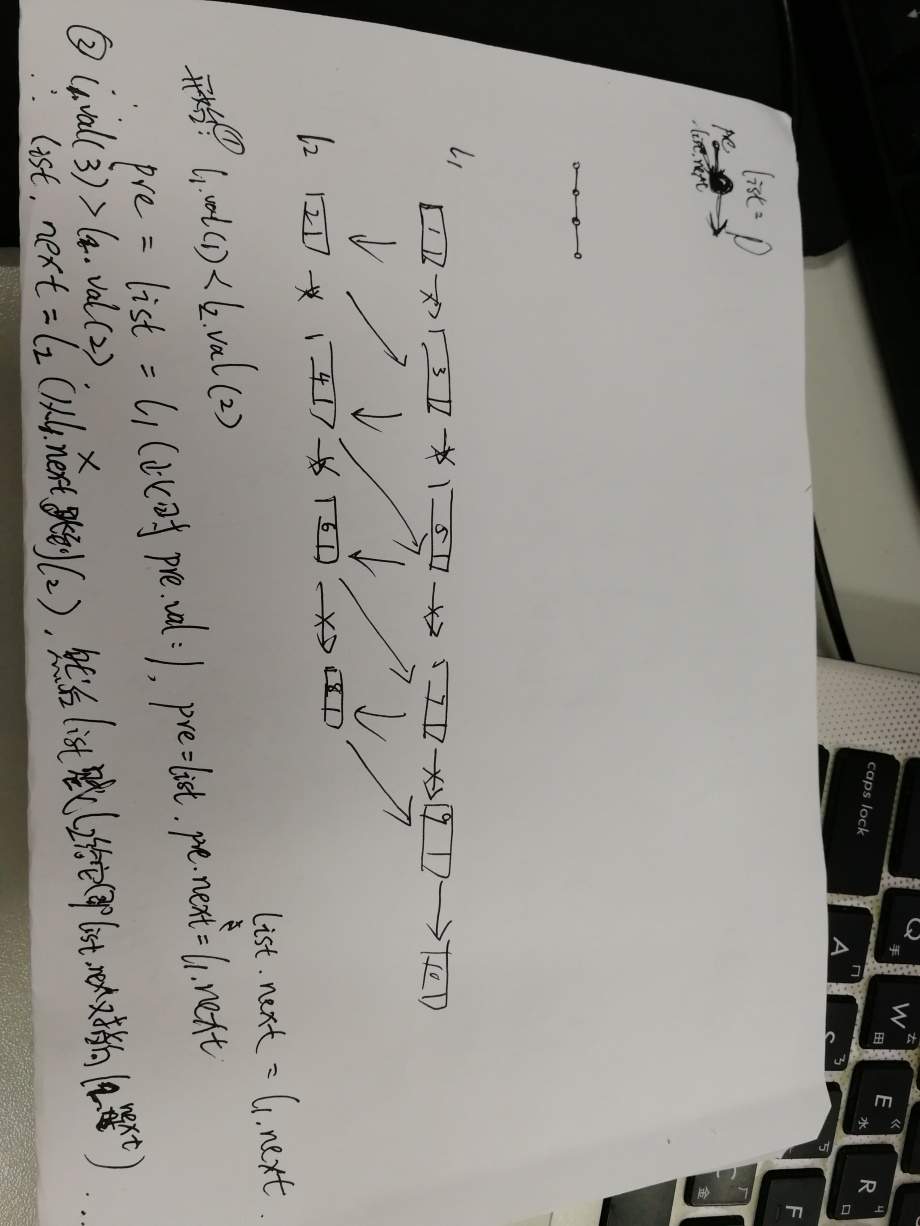
输入两个单调递增的链表，输出两个链表合成后的链表，当然我们需要合成后的链表满足单调不减规则。

思路：

如果pre和list都是null，先用pre存进第一个取得的节点。然后让它等于list，这样之后list指向的地址怎么变化，它的指向就会跟着list去跳了。

其中两句标红的是关键，



**package** facehandjava.Linked;  
  
**public class** MergeTwo {  
 **public static void** main(String[] args) {  
  
 Node n6 = **new** Node(11,**null**);  
 Node n5 = **new** Node(9, n6);  
 Node n4 = **new** Node(7, n5);  
 Node n3 = **new** Node(5, n4);  
 Node n2 = **new** Node(3, n3);  
 Node n1 = **new** Node(1, n2);  
  
 Node n33 = **new** Node(33, **null**);  
 Node n22 = **new** Node(3, n33);  
 Node n11 = **new** Node(2, n22);  
  
 Node mergeTwo = *MergeTwo*(n1, n11);  
 **while** (mergeTwo != **null**) {  
 System.***out***.print(mergeTwo.getVal()+**","**);  
 mergeTwo = mergeTwo.getNext();  
 }  
 }  
  
 **public static** Node MergeTwo(Node n1, Node n2) {  
 Node l = **null**;*//临时* Node MT = **null**;*//保存所有* **while** (n1 != **null** && n2 != **null**) {  
 **if** (n1.getVal() >= n2.getVal()) {  
 **if** (l == **null**) {  
 MT = l = n2;  
 n2 = n2.getNext();  
 }**else** {  
 l.setNext(n2);  
 l = l.getNext();  
 n2 = n2.getNext();  
 }  
 }**else** {  
 **if** (l == **null**) {  
 MT = l = n1;  
 n1 = n1.getNext();  
 } **else** {  
 l.setNext(n1);  
 l = l.getNext();  
 n1 = n1.getNext();  
 }  
 }  
 }  
 **if** (n1 == **null**) {  
 l.setNext(n2);  
 }  
 **if** (n2 == **null**) {  
 l.setNext(n1);  
 }  
 **return** MT;  
 }  
}