Task 01  
public class Solution {  
 public ListNode getIntersectionNode(ListNode headA, ListNode heads) {  
  
 if (headA= null || head8==null) {  
  
 return null;  
  
 }  
  
 ListNode CA=headA;  
  
 ListNode CB=headB;  
  
 while(CA != CB) {  
  
 CA=(CA==null)? headb:CA.next;  
  
 CB=(CB==null)? headA:CB.next;  
  
 }  
 return CA;  
 }  
   
 Task 02  
 class Solution {  
 public ListNode deleteDuplicates(ListNode head) {  
  
 ListNode current=head;  
  
 while(current != null && current.next!=null){  
  
 if (current.val= current.next.val){  
  
 current.next=current.next.next;  
  
 } else{  
  
 current=current.next;  
 }  
 return head;  
  
  
 Task 03  
  
 class Solution {  
 public ListNode mergeTwoLists(ListNode list1, ListNode list2) {  
 ListNode list= new ListNode(-1);  
 Listhode current = list;  
 while (list == null && list2!= null) {  
 if (list.vel == list2.val) {  
current.next=lists; lists=list.next;  
} else {  
current.next=list2; list2=list2.next;  
}  
current=current.next;  
if (list != null) {  
 current.next list;  
if (list2 != null) {  
current.next list2;  
return list.next;

task 04

class Solution {  
public ListNode addTwoNumbers(ListNode l1, ListNode l2) {  
ListNode list=new ListNode(0);  
ListNode current=list;  
int carry=0;  
while(11 != null || 12 != null || carry !=0){  
int val1=(11!= null)? 11.val:0;  
int val2=(12 != null) 12.val:0;  
Int sum=val1+val2+carry;  
carry sum/10;  
int digit=sum%10;  
current.next=new ListNode(digit);  
current=current.next;  
if(11 != null){  
11=11.next;  
}  
1(12 != null) {  
12=12.next;  
return list.next;