Write a program to find the node at which the intersection of two singly linked lists begins.

For example, the following two linked lists:

[](https://assets.leetcode.com/uploads/2018/12/13/160_statement.png)

begin to intersect at node c1.

问你检查有没有焦点，如果有return，，没有就output null

思路：

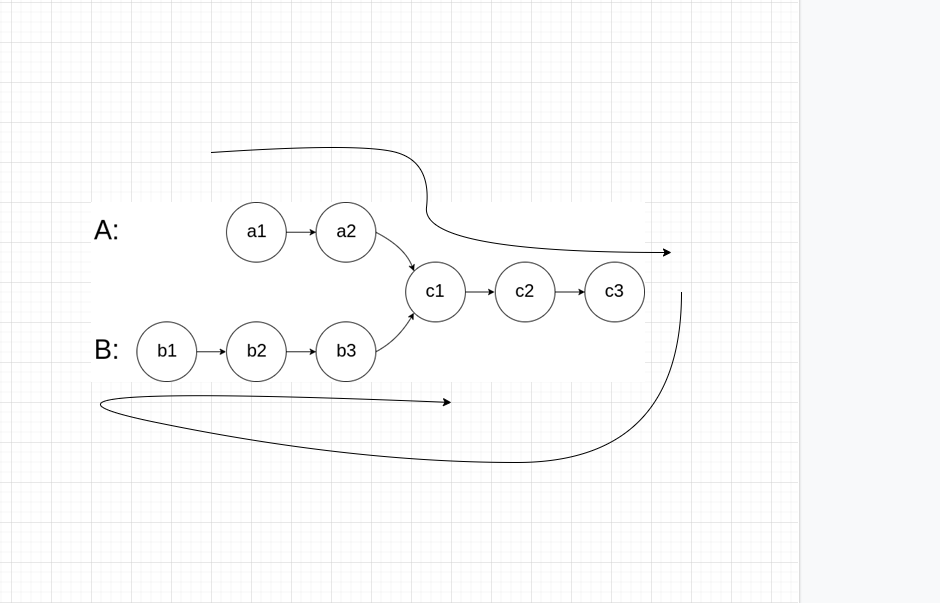
基本方案：

分别遍历一遍，记录两个Linkedlist长度

然后让长的那个先走 （长-短）步，然后一起走，那么就会在焦点碰头

高级方案：

一起走，走到头（本身=null，而不是next=null），就换另一边的linked list走，经过两轮以后，要么是焦点，要么是尾端Null，因为两个pointer走过的路程是一样的



假设从a1开始走到c3,然后c3.next，下一个循环发现自己是Null，走b1,b2,b3,c1

那么一共会走上面的长度+c的长度+b的长度

同理下面的pointer会走下面的长度+c的长度+a的长度

长度相等，在c1碰头

如果没有交叉

就是a1的长度加b1的长度，这时两个人正好都是null ，return null

/\*\*

\* Definition for singly-linked list.

\* public class ListNode {

\* int val;

\* ListNode next;

\* ListNode(int x) {

\* val = x;

\* next = null;

\* }

\* }

\*/

public class Solution {

public ListNode getIntersectionNode(ListNode headA, ListNode headB) {

ListNode pointer\_a=headA;

ListNode pointer\_b=headB;

while(pointer\_a!=pointer\_b){

if(pointer\_a!=null) pointer\_a=pointer\_a.next; //注意如果没交叉，是可以存储null一个循环的，因为假设他到了最后一个，那么经过这句话它会变成null，而B也会变成null，这样在while判断就会跳出去

else pointer\_a=headB;

if(pointer\_b!=null) pointer\_b=pointer\_b.next;

else pointer\_b=headA;

}

return pointer\_a;

}

}