Linked List Cycle II

 Given a linked list, return the node where the cycle begins. If there is no cycle, return null.

Follow up:  
Can you solve it without using extra space?

大概思路就是：两个指针ptr1和ptr2，都从链表头开始走，ptr1每次走一步，ptr2每次走两步，等两个指针重合时，就说明有环，否则没有。如果有环的话，那么让ptr1指向链表头，ptr2不动，两个指针每次都走一步，当它们重合时，所指向的节点就是环开始的节点。

证明如下：我们知道ptr2每次都要比ptr1多走一步。

现在，假设ptr1需要m步第一次走到环的开始节点，那么ptr2应该走到了环的第m个位置（注意：这里是指环里的位置，ptr1在0处，ptr2在m处）。ptr2每次都比ptr1多走一步且它们都在环内，所以需要n-m步（n是环的大小），才能追上ptr1，达到重合。这个重合位置是[2\*（n-m）+m-x\*n]%n=n-m（因为ptr1从环内0开始走，走n – m步），这样我们就可以知道，重合的位置距离环开始节点有m步，head距离环开始节点也是m步。