142. Linked List Cycle II

题目描述：https://leetcode.com/problems/linked-list-cycle-ii/

> 给定一个链表，如果这个链表有环，找到环的起始点，无环则返回NULL

解题思路：

Slow = head;
Fast = head;
假设Slow走了k步， 则Fast走了2 * k步
如果Slow 和 Fast相遇，则 2 * k - k = n * C (C是环的周长)
因此k = n * C，只要走C的整数倍两个指针就可以相遇一次。
如果此时有一个从head开始走的指针P。
如果要走到环始点P需要走L步，如果这个时候Slow也跟着走，那么它就走了k+L步 = n * C + L
如果Slow走n * C + L步，则证明它返回到了环始点，这时候P和Slow重合。

代码：

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution {
public:
    ListNode *detectCycle(ListNode *head) {
        if(head == NULL || head->next == NULL){
            return NULL;
        }
        ListNode *s, *f;
        s = head;
        f = head;
        while(s && f){
            if(f->next == NULL){
               return NULL;
            }
            s = s->next;
            f = f->next->next;
            if(s == f){
                break;
            }
        }
        if(s == NULL || f == NULL){
            return NULL;
        }
        ListNode *p = head;
        while(p != s){
            p = p->next;
            s = s->next;
        }
        return p;
    }
};
```