142. Linked List Cycle II

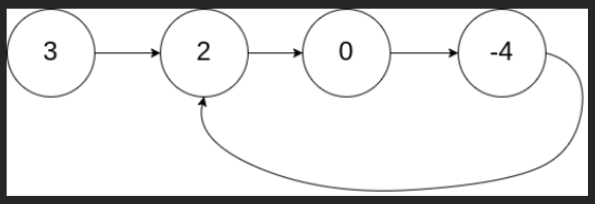
Medium | Topics | Companies

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

There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the `next` pointer. Internally, `pos` is used to denote the index of the node that tail's `next` pointer is connected to (**0-indexed**). It is `-1` if there is no cycle. **Note that** `pos` **is not passed as a parameter**.

Do not modify the linked list.

Example 1:



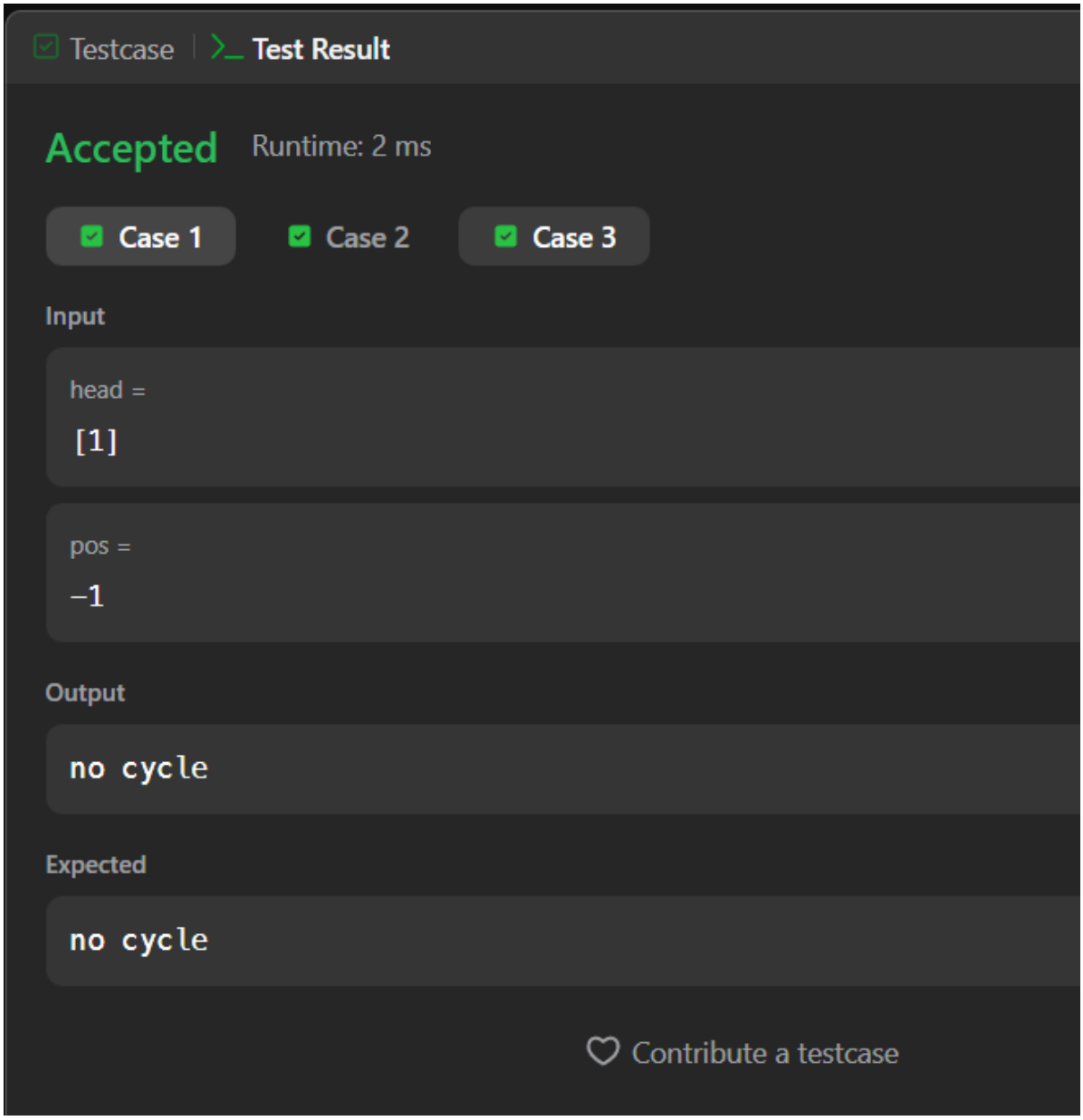

```
Input: head = [3,2,0,-4], pos = 1
Output: tail connects to node index 1
Explanation: There is a cycle in the linked list, where tail connects to the second node.
```

Code

C | Auto

```c
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */
struct ListNode *detectCycle(struct ListNode *head) {
    struct ListNode *f=head,*s=head;
    int loop=0;
    if(f!=NULL && f->next!=NULL){
        while(f!=NULL && f->next !=NULL){
            s=s->next;
            f=f->next->next;
            if(f==s){
                loop=1;
                break;
            }
        }
        if(loop){
        f=head;
        while(f!=s){
            f=f->next;
            s=s->next;
        }
        return f;
    }

    }return NULL;
}
```


Testcase
Test Result
Accepted
Runtime: 2 ms
Case 1
Case 2
Case 3
Input
head =
[1]
pos =
-1
Output
no cycle
Expected
no cycle
Contribute a testcase