

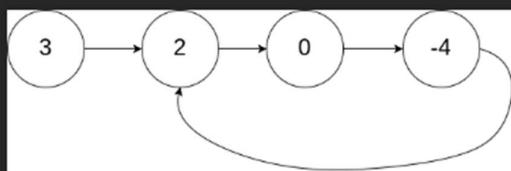
Given `head`, the head of a linked list, determine if the linked list has a cycle in it.

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. **Note that `pos` is not passed as a parameter.**

Return `true` if there is a cycle in the linked list. Otherwise, return `false`.

*

Example 1:



Input: `head = [3,2,0,-4]`, `pos = 1`

Output: `true`

Explanation: There is a cycle in the linked list, where the tail connects to the 1st node (0-indexed).

```
1 /**
2  * Definition for singly-linked list.
3  * struct ListNode {
4  *     int val;
5  *     struct ListNode *next;
6  * };
7 */
8
9 bool hasCycle(struct ListNode *head) {
10    if (head == NULL || head->next == NULL)
11        return false;
12
13    struct ListNode *slow = head;
14    struct ListNode *fast = head;
15
16    while (fast != NULL && fast->next != NULL) {
17        slow = slow->next;           // move slow by 1
18        fast = fast->next->next;    // move fast by 2
19
20        if (slow == fast) {          // they meet → cycle
21            return true;
22        }
23    }
24    return false;                  // fast reached NULL → no cycle
25}
```

Accepted Runtime: 3 ms

Case 1 Case 2 Case 3

Input

```
head =  
[3,2,0,-4]
```

```
pos =  
1
```

Output

```
true
```

Expected

```
true
```

•

Accepted Runtime: 3 ms

Case 1 Case 2 Case 3

Input

```
head =  
[1,2]
```

```
pos =  
0
```

Output

```
true
```

Expected

```
true
```

Accepted

Runtime: 3 ms

Case 1

Case 2

Case 3

Input

head =

[1]

pos =

-1

Output

false

Expected

false