

## LAB\_07B\_LEETCODE\_141. Linked List Cycle

```
bool hasCycle(struct ListNode *head) {  
  
    if (head == NULL || head->next == NULL) {  
        return false;  
    }  
  
    struct ListNode *slow = head;  
    struct ListNode *fast = head;  
  
    while (fast != NULL && fast->next != NULL) {  
  
        slow = slow->next;  
  
        fast = fast->next->next;  
  
        if (slow == fast) {  
            return true;  
        }  
    }  
    return false;  
}
```

The screenshot shows a LeetCode problem page for "Linked List Cycle". The submission was accepted by SHRIHARI VISWANATHAN on Dec 08, 2025 at 19:10. The runtime distribution chart indicates that 0.3% of solutions used 12 ms of runtime, which is highlighted in blue. The code editor displays the C implementation provided above.

```
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
11     if (head == NULL || head->next == NULL) {
12         return false;
13     }
14
15     struct ListNode *slow = head;
16     struct ListNode *fast = head;
17
18     while (fast != NULL && fast->next != NULL) {
19
20         slow = slow->next;
21
22         fast = fast->next->next;
23
24         if (slow == fast) {
25             return true;
26         }
27     }
28
29     // If the loop completes, 'fast' reached the end, meaning no cycle was found.
30     return false;
31 }
```