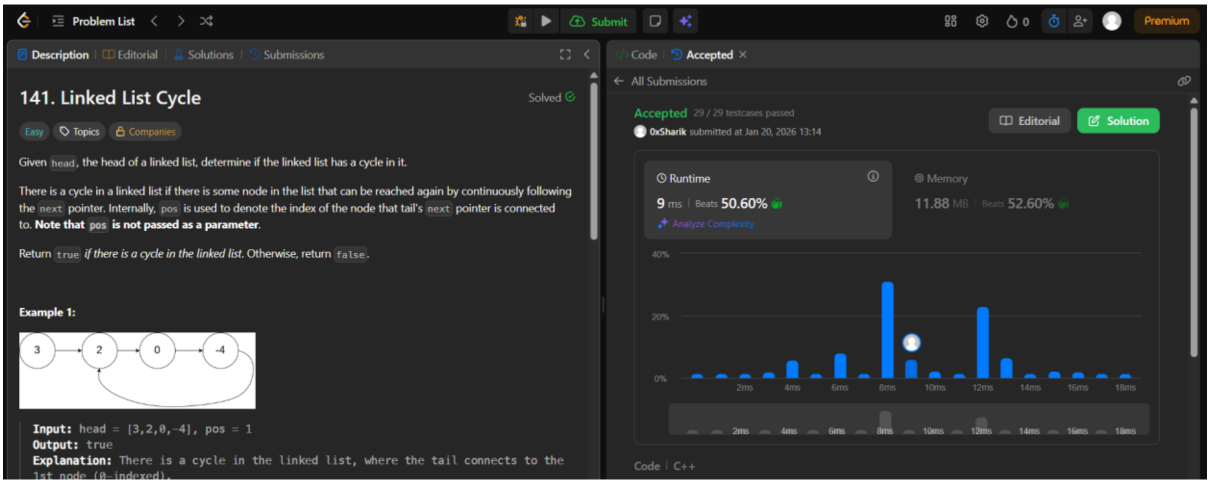
PROBLEM 1



141. Linked List Cycle

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).
```

Accepted 29 / 29 testcases passed

0xSharik submitted at Jan 20, 2026 13:14

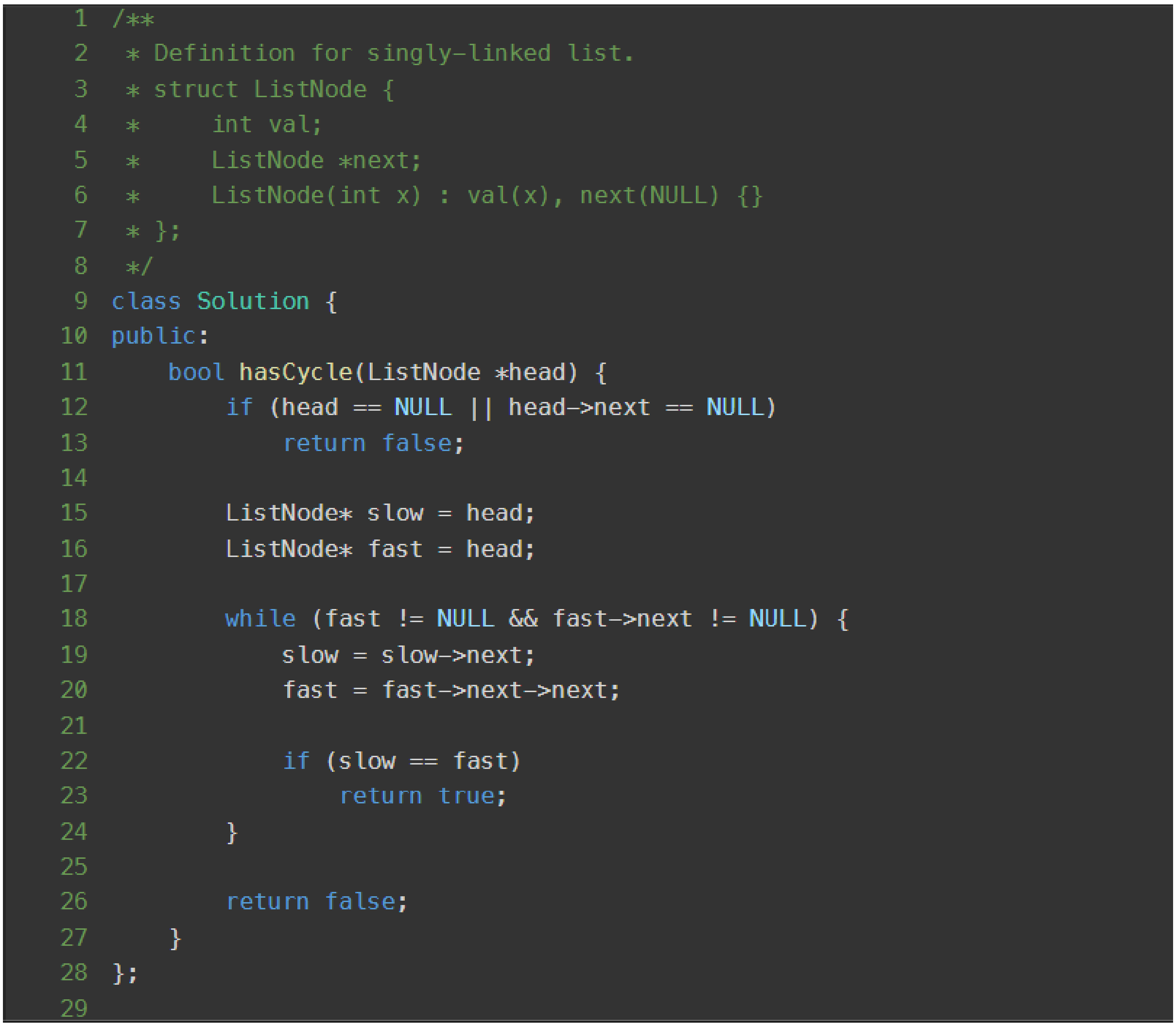
Runtime

9 ms | Beats 50.60%

Memory

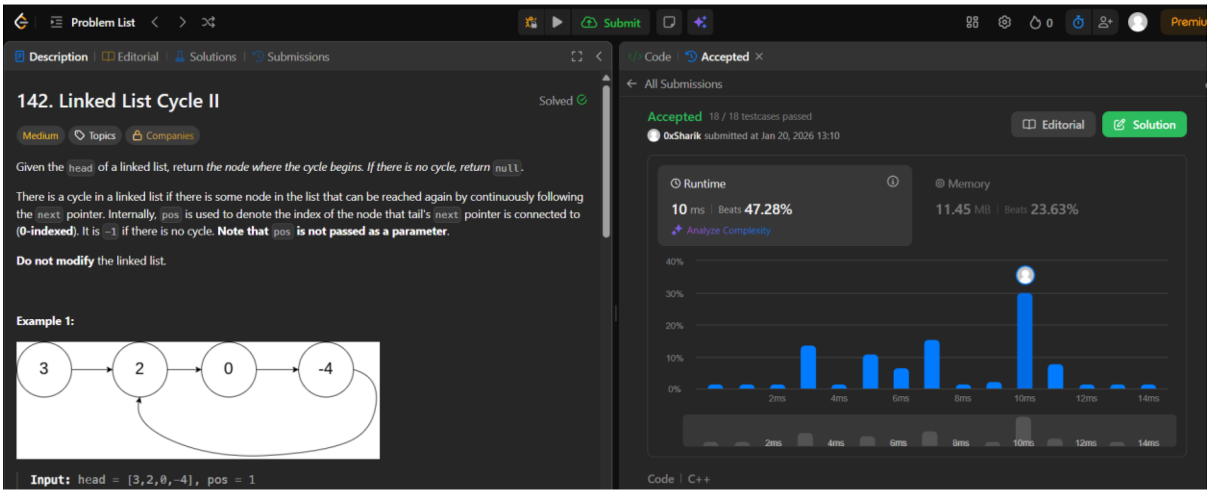
11.88 MB | Beats 52.60%

Code | C++



```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution {
public:
    bool hasCycle(ListNode *head) {
        if (head == NULL || head->next == NULL)
            return false;

        ListNode* slow = head;
        ListNode* fast = head;

        while (fast != NULL && fast->next != NULL) {
            slow = slow->next;
            fast = fast->next->next;

            if (slow == fast)
                return true;
        }

        return false;
    }
};

```

PROBLEM 2



142. Linked List Cycle II

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:

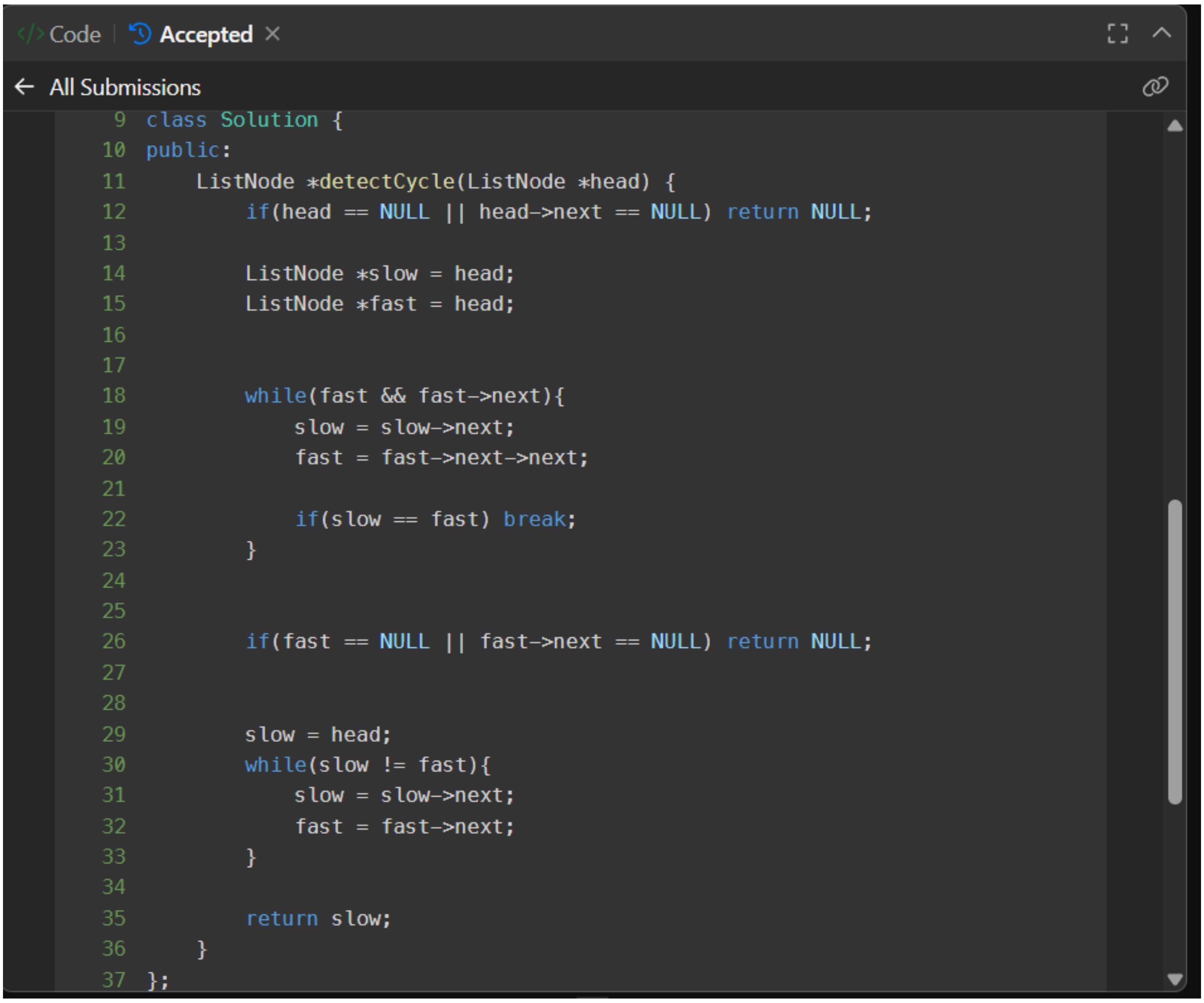
Input: `head = [3,2,0,-4], pos = 1`

Accepted 18 / 18 testcases passed

0xSharik submitted at Jan 20, 2026 13:10

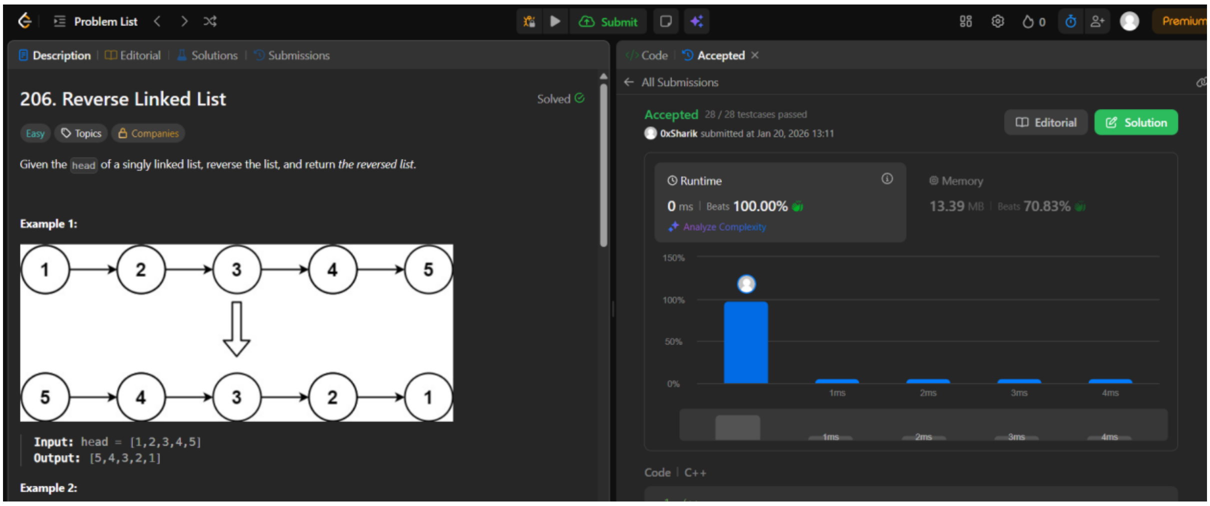
10 ms | Beats 47.28%

11.45 MB | Beats 23.63%



```
class Solution {
public:
    ListNode *detectCycle(ListNode *head) {
        if(head == NULL || head->next == NULL) return NULL;

        ListNode *slow = head;
        ListNode *fast = head;


        while(fast && fast->next){
            slow = slow->next;
            fast = fast->next->next;

            if(slow == fast) break;
        }


        if(fast == NULL || fast->next == NULL) return NULL;


        slow = head;
        while(slow != fast){
            slow = slow->next;
            fast = fast->next;
        }

        return slow;
    }
};
```

PROBLEM 3



206. Reverse Linked List

Given the `head` of a singly linked list, reverse the list, and return *the reversed list.*

Example 1:

```
Input: head = [1,2,3,4,5]
Output: [5,4,3,2,1]
```

Example 2:

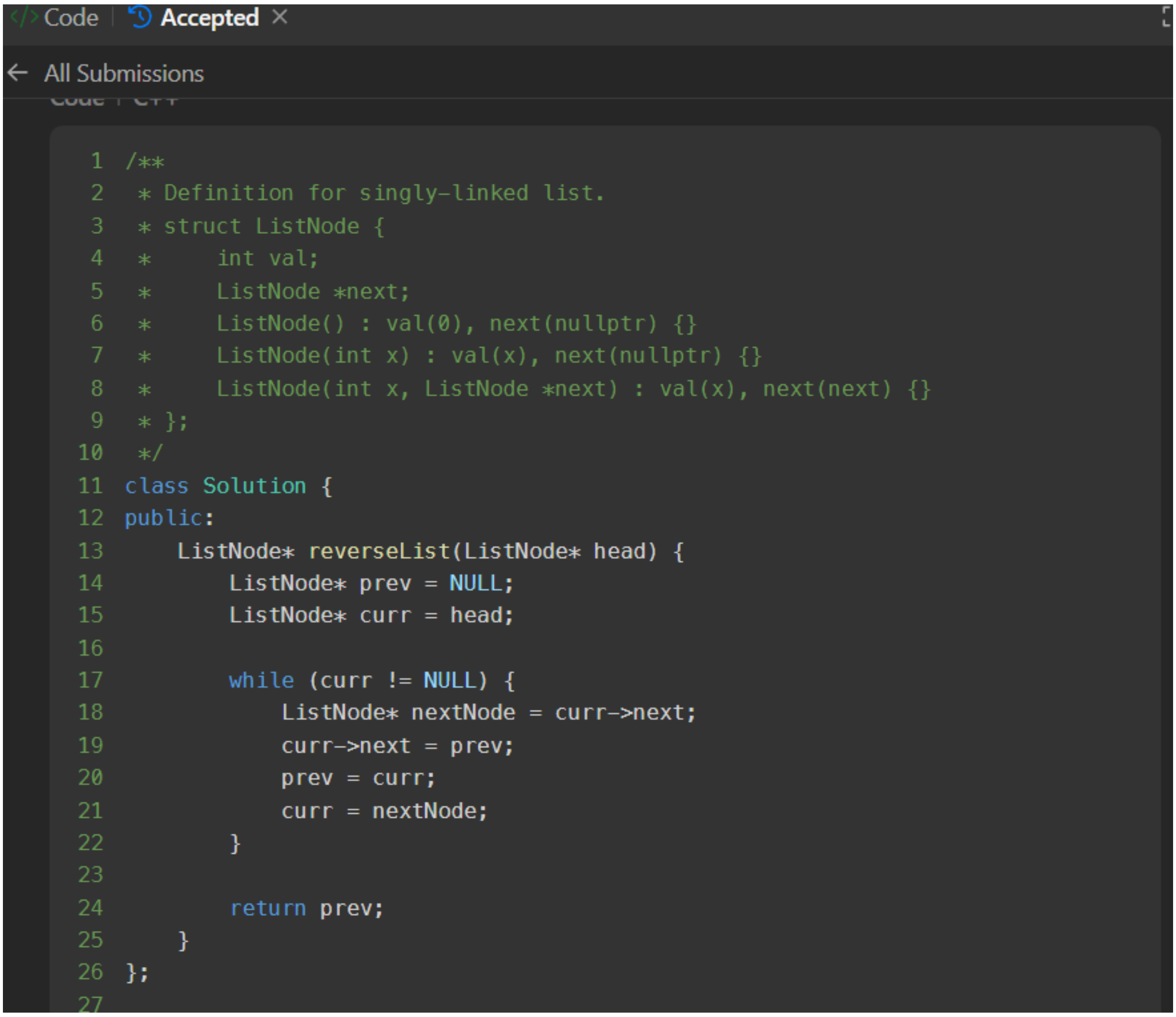
Accepted 28 / 28 testcases passed

0xSharik submitted at Jan 20, 2026 13:11

Runtime 0 ms | Beats 100.00%

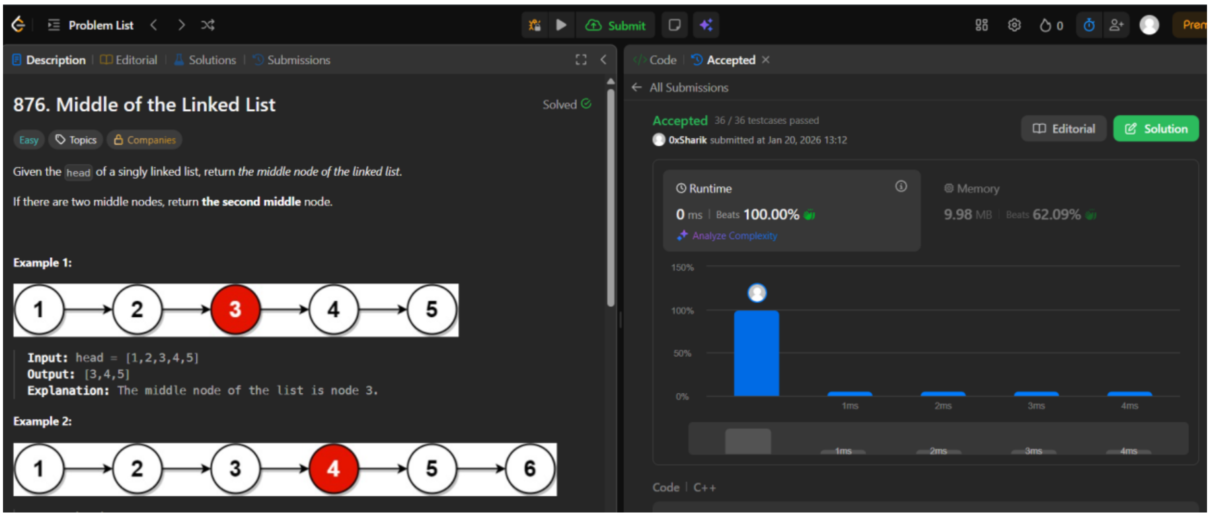
Memory 13.39 MB | Beats 70.83%

Code | C++



```cpp
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* reverseList(ListNode* head) {
        ListNode* prev = NULL;
        ListNode* curr = head;

        while (curr != NULL) {
            ListNode* nextNode = curr->next;
            curr->next = prev;
            prev = curr;
            curr = nextNode;
        }

        return prev;
    }
};
```

PROBLEM 4



876. Middle of the Linked List

Given the head of a singly linked list, return *the middle node of the linked list*.

If there are two middle nodes, return **the second middle** node.

Example 1:

```
Input: head = [1,2,3,4,5]
Output: [3,4,5]
Explanation: The middle node of the list is node 3.
```

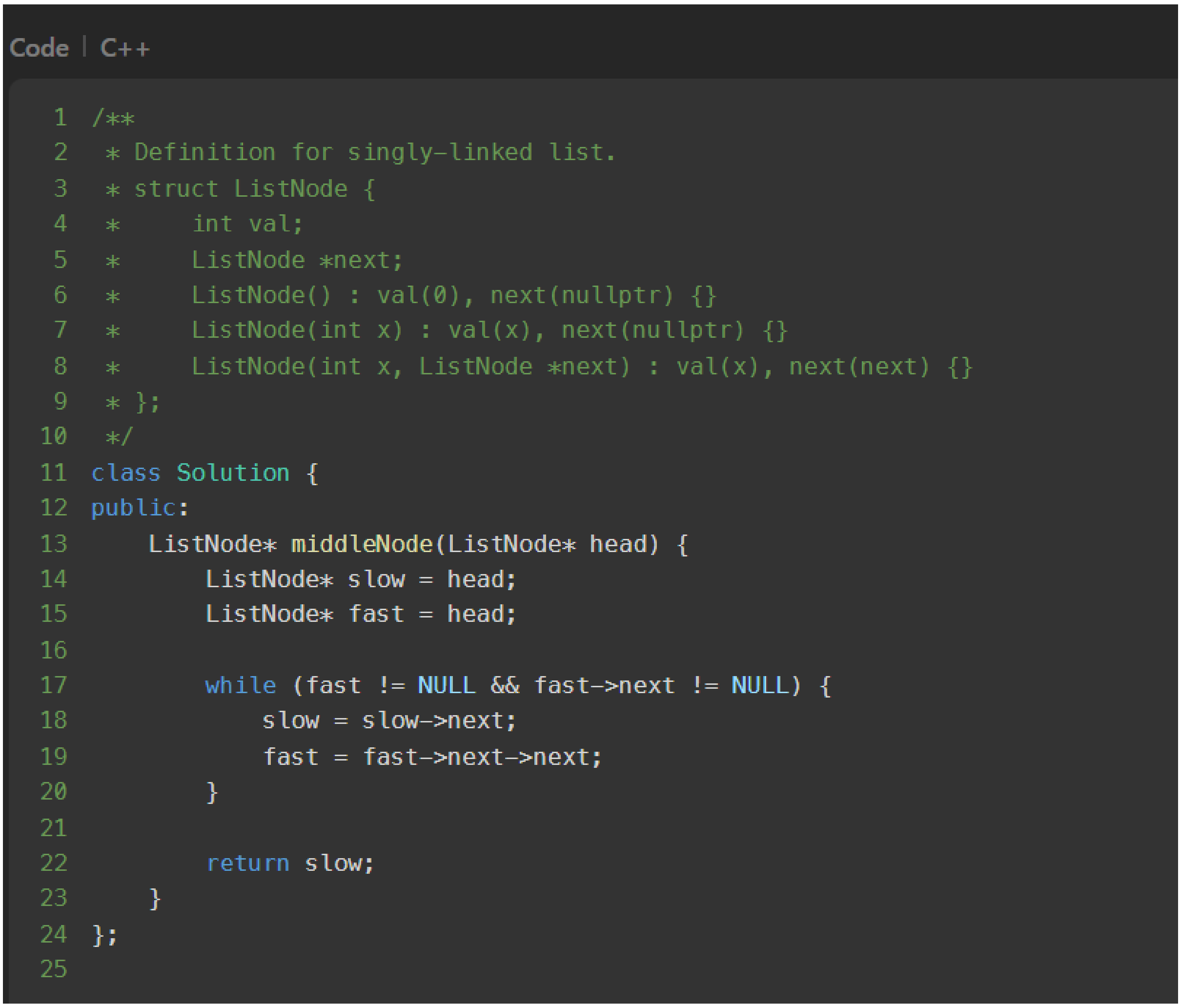
Example 2:

Accepted 36 / 36 testcases passed

0xSharik submitted at Jan 20, 2026 13:12

Runtime 0 ms | Beats 100.00%

Memory 9.98 MB | Beats 62.09%



Code | C++

```cpp
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* middleNode(ListNode* head) {
        ListNode* slow = head;
        ListNode* fast = head;

        while (fast != NULL && fast->next != NULL) {
            slow = slow->next;
            fast = fast->next->next;
        }

        return slow;
    }
};

```