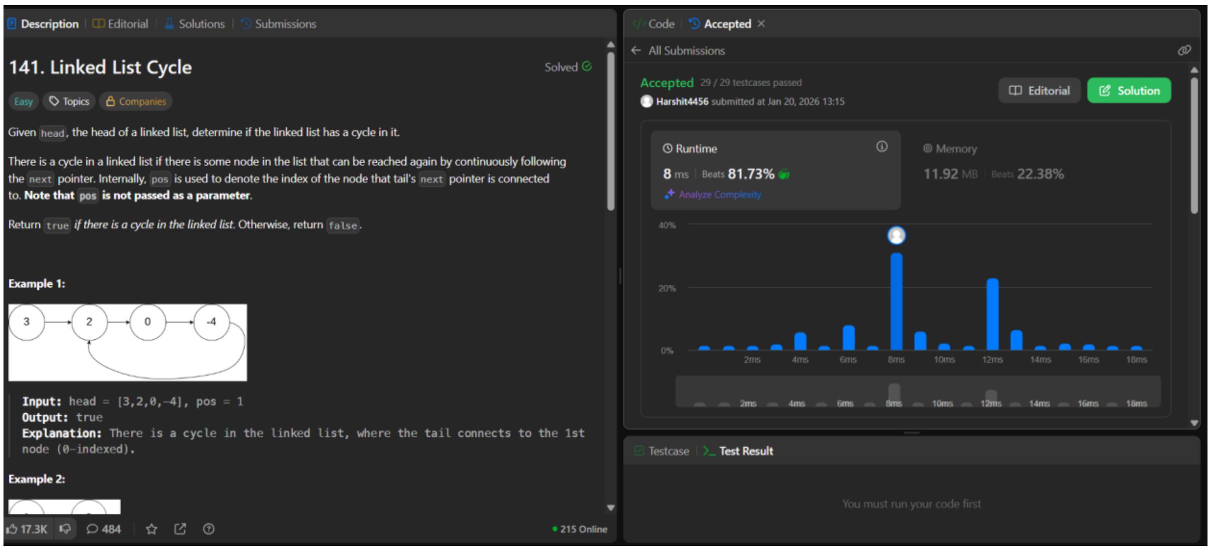
PROBLEM 141



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).
```

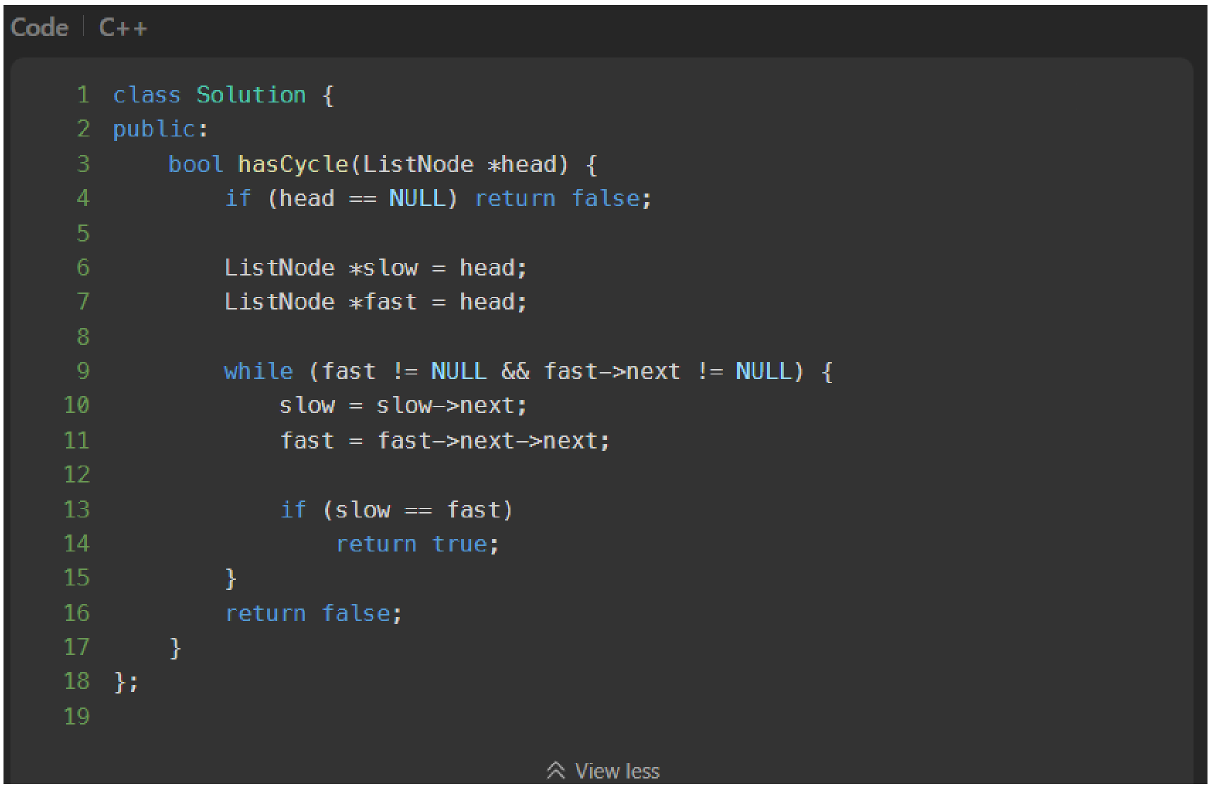
Example 2:

Accepted 29 / 29 testcases passed
Harshit4456 submitted at Jan 20, 2026 13:15

Runtime 8 ms | Beats 81.73%

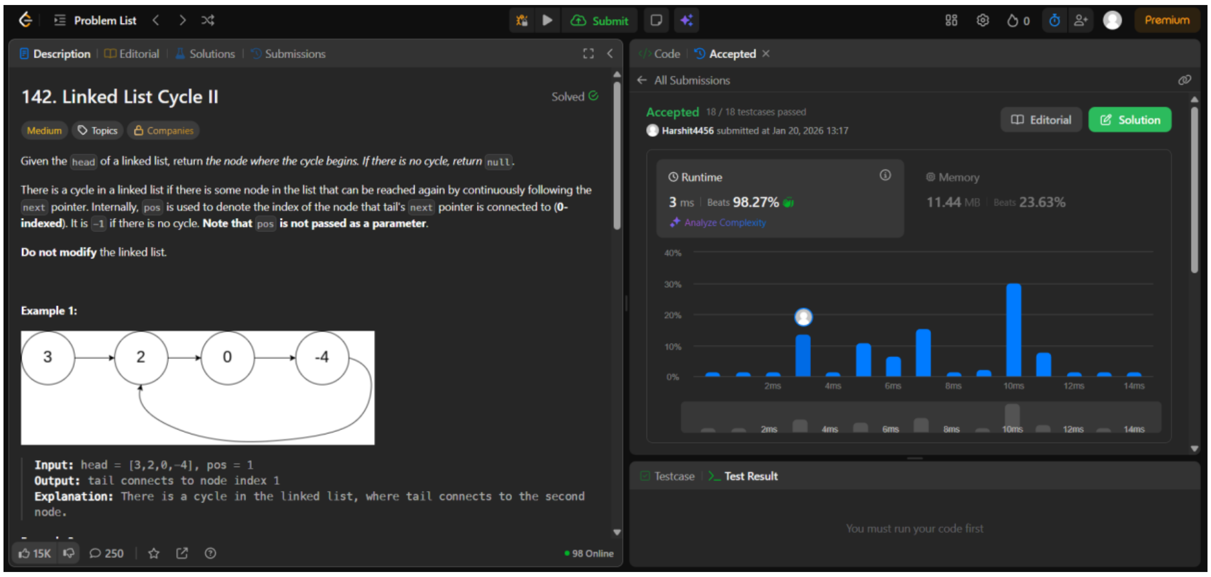
Memory 11.92 MB | Beats 22.38%

You must run your code first



```cpp
class Solution {
public:
    bool hasCycle(ListNode *head) {
        if (head == 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 142



142. Linked List Cycle II

Solved

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:

$$\begin{tikzcd} 3 \arrow[r] & 2 \arrow[r] & 0 \arrow[r] & -4 \arrow[ll, bend left] \end{tikzcd}$$

```
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.
```

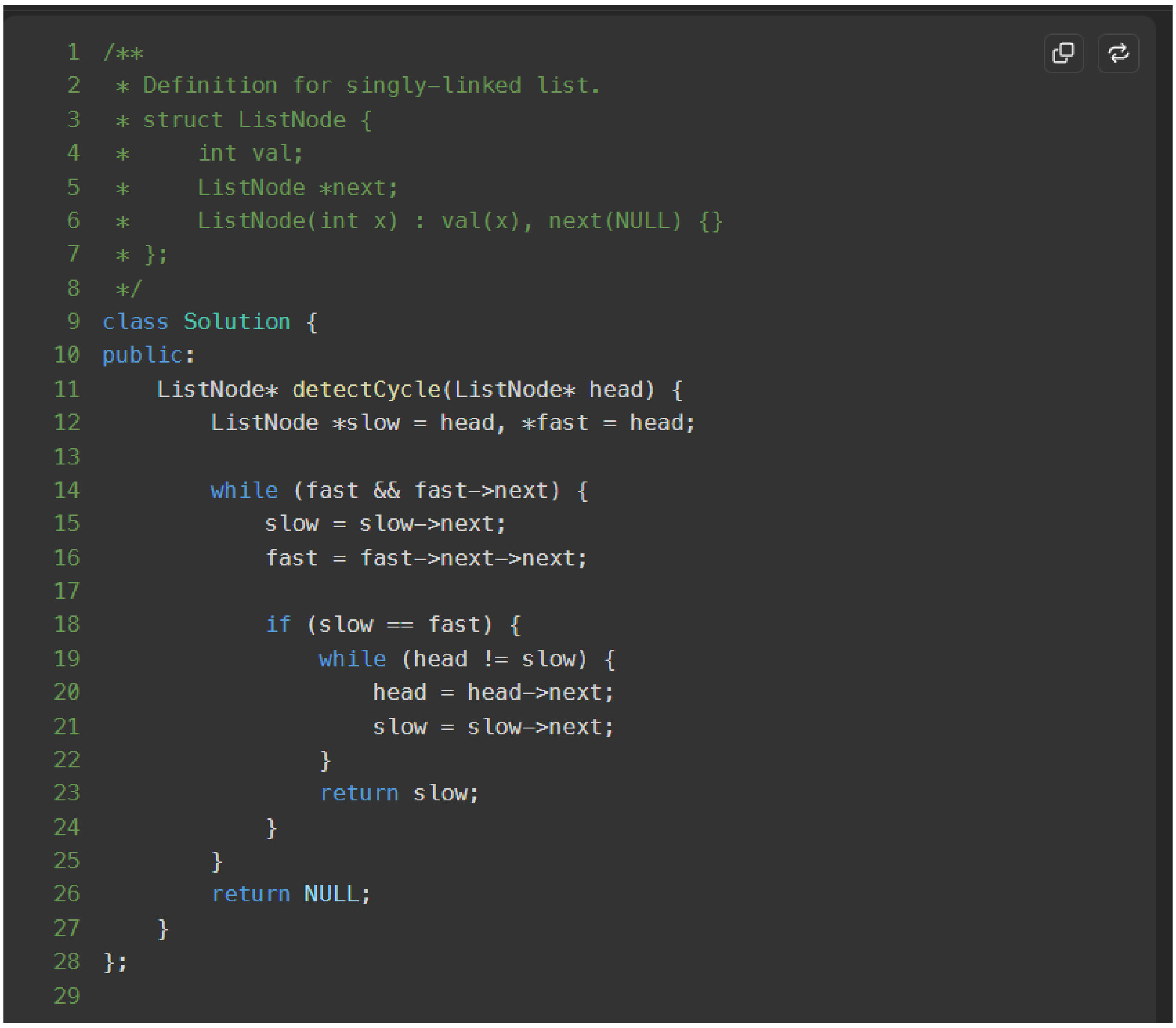
Accepted 18 / 18 testcases passed

Harshit4456 submitted at Jan 20, 2026 13:17

Runtime
3 ms | Beats 98.27%

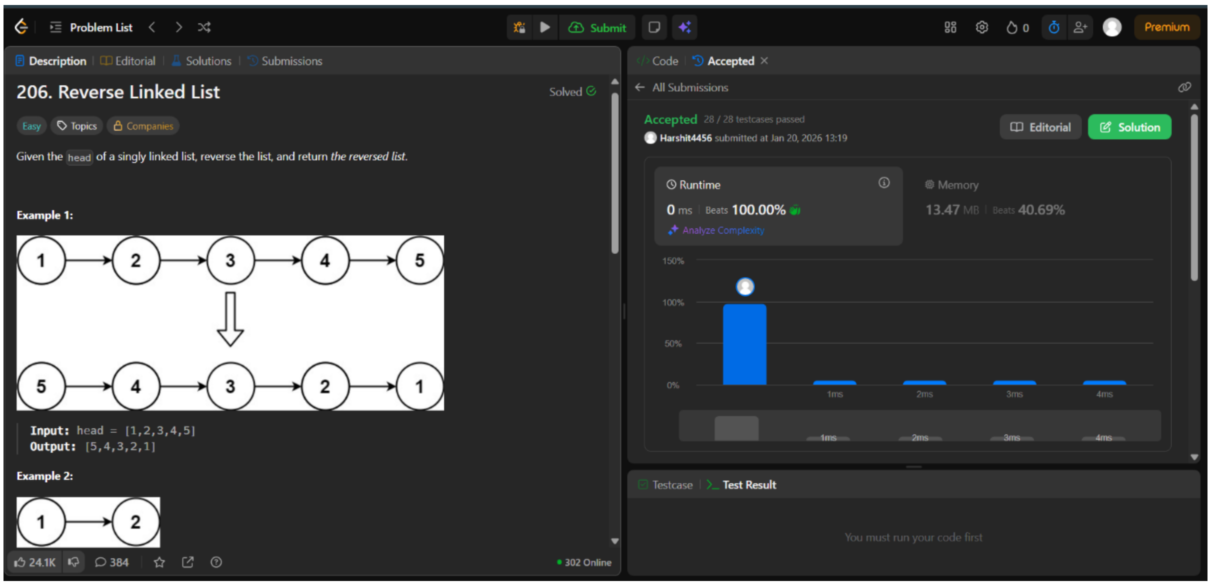
Memory
11.44 MB | Beats 23.63%

You must run your code first



```
/**
 * 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) {
        ListNode *slow = head, *fast = head;

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

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

```

PROBLEM 206



206. Reverse Linked List

Solved

Easy · Topics · Companies

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:

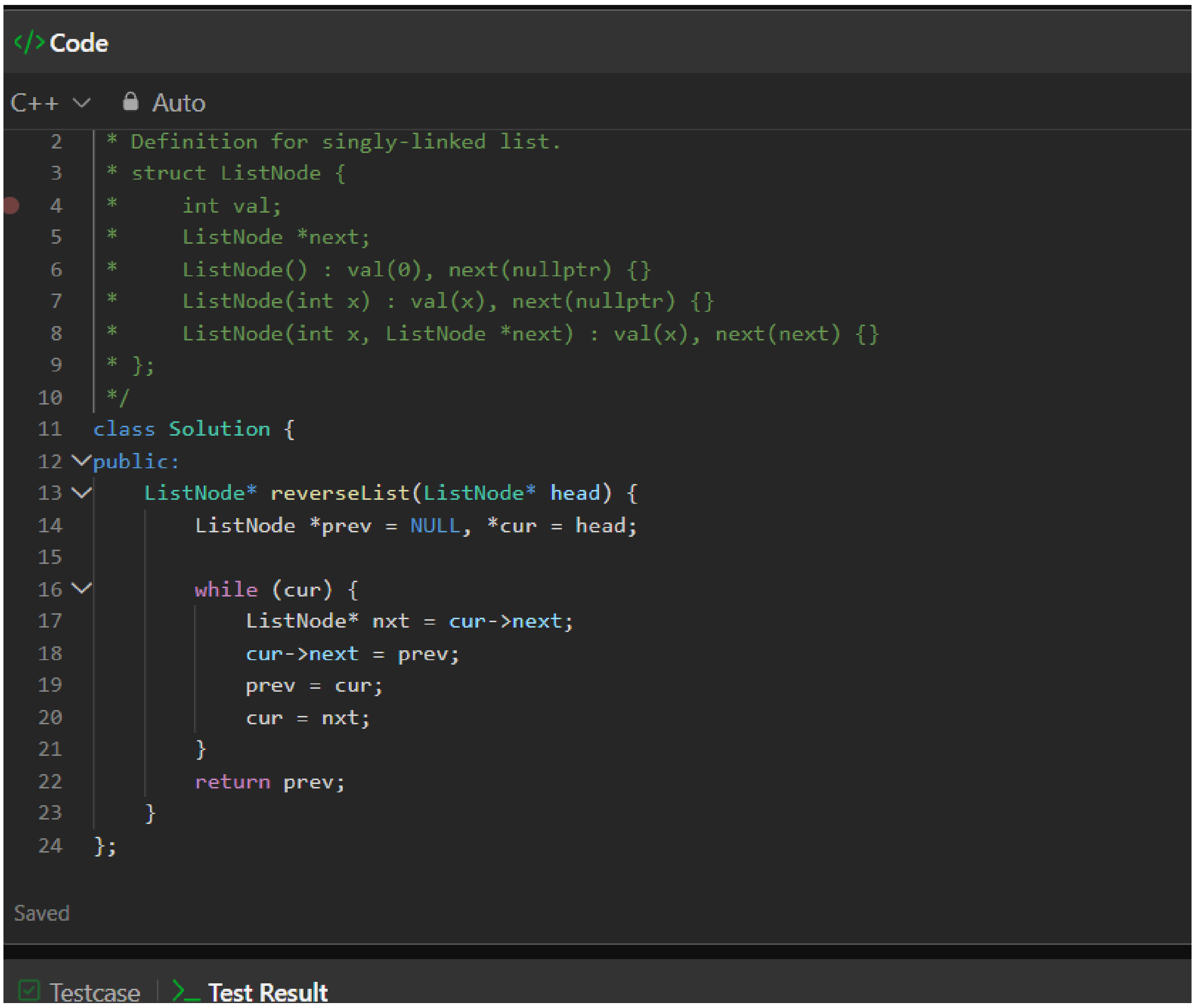
Accepted 28 / 28 testcases passed

Harshit4456 submitted at Jan 20, 2026 13:19

Runtime
0 ms | Beats 100.00%
Analyze Complexity

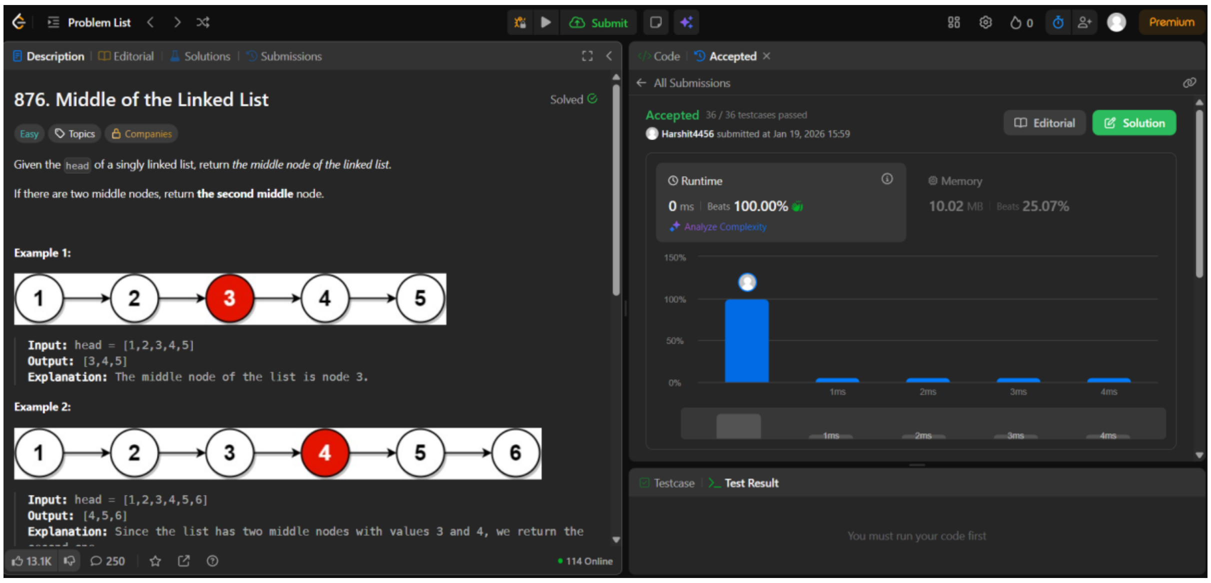
Memory
13.47 MB | Beats 40.69%

You must run your code first



```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, *cur = head;

        while (cur) {
            ListNode* nxt = cur->next;
            cur->next = prev;
            prev = cur;
            cur = nxt;
        }
        return prev;
    }
};
```

PROBLEM 876



876. Middle of the Linked List

Solved

Easy · Topics · Companies

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:

```
Input: head = [1,2,3,4,5,6]
Output: [4,5,6]
Explanation: Since the list has two middle nodes with values 3 and 4, we return the
```

13.1K · 250 · 114 Online

Accepted 36 / 36 testcases passed

Harshit4456 submitted at Jan 19, 2026 15:59

Runtime

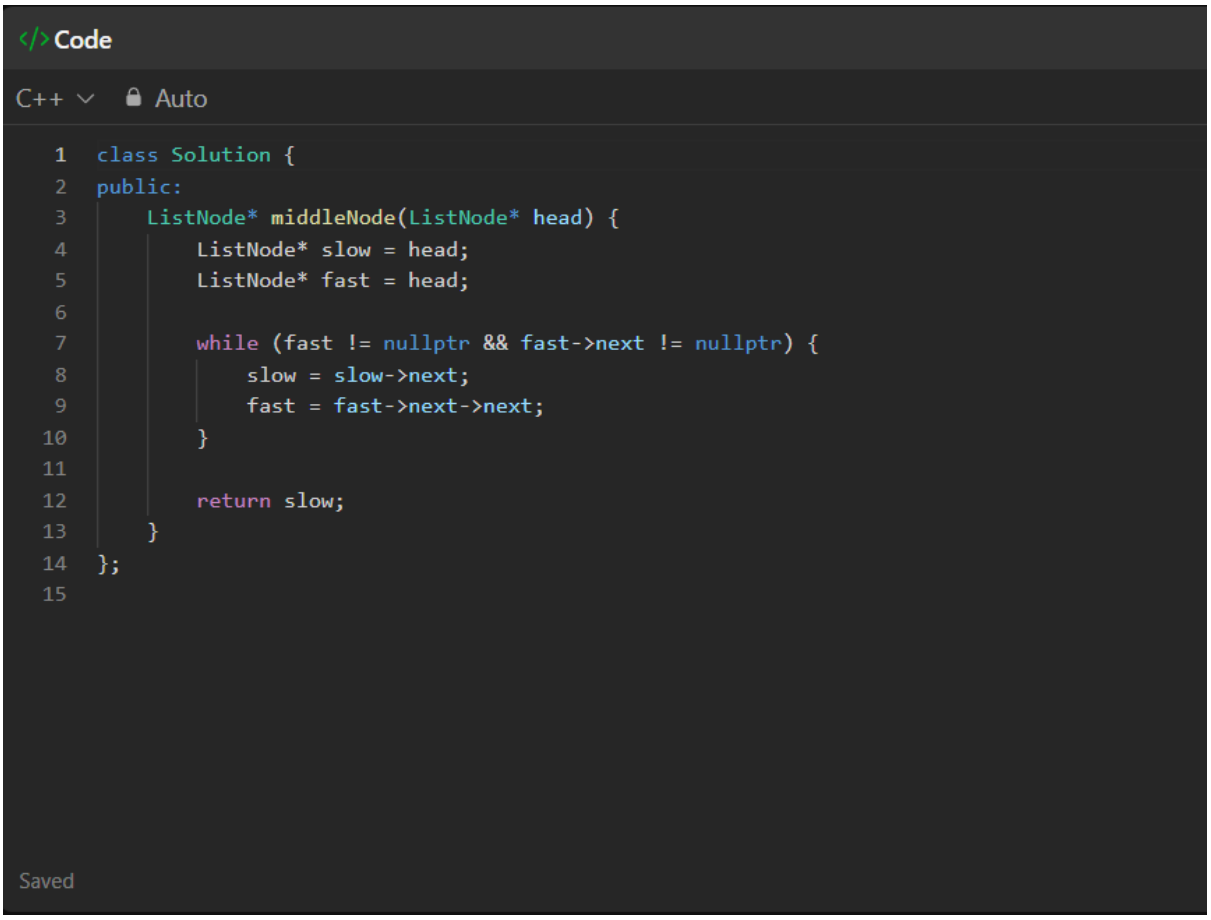
0 ms | Beats 100.00%

Analyze Complexity

Memory

10.02 MB | Beats 25.07%

You must run your code first



</> Code

C++ · Auto

```cpp
class Solution {
public:
    ListNode* middleNode(ListNode* head) {
        ListNode* slow = head;
        ListNode* fast = head;

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

        return slow;
    }
};

```

Saved