

LEET CODE 876:

Screenshot of LeetCode problem 876: Middle of the Linked List.

The page header includes navigation icons and links for Description, Editorial, Solutions, and Submissions.

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:

```
graph LR; 1((1)) --> 2((2)); 2 --> 3((3)); 3 --> 4((4)); 4 --> 5((5));
```

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

Example 2:

```
graph LR; 1((1)) --> 2((2)); 2 --> 3((3)); 3 --> 4((4)); 4 --> 5((5)); 5 --> 6((6));
```

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 second one.

Screenshot of a code editor showing the solution for LeetCode problem 876.

The code is written in C and implements the two-pointer (slow and fast) technique to find the middle node of a singly linked list.

```
1 struct ListNode* middleNode(struct ListNode* head) {
2     struct ListNode* slow = head;
3     struct ListNode* fast = head;
4
5     while (fast != NULL && fast->next != NULL) {
6         slow = slow->next;
7         fast = fast->next->next;
8     }
9
10    return slow;
11}
12
```

Problem List < >

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

```
head =
[1,2,3,4,5]
```

Output

```
[3,4,5]
```

Expected

```
[3,4,5]
```

Contribute a testcase

Problem List < >

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

```
head =
[1,2,3,4,5,6]
```

Output

```
[4,5,6]
```

Expected

```
[4,5,6]
```

Contribute a testcase
