

## Middle Of The Linked List

```
struct ListNode* middleNode(struct ListNode* head) {  
    struct ListNode* slow = head;  
    struct ListNode* fast = head;  
  
    while (fast != NULL && fast->next != NULL) {  
        slow = slow->next;  
        fast = fast->next->next;  
    }  
  
    return slow;  
}
```

## Output

**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:**

Diagram: 1 → 2 → 3 → 4 → 5 (Node 3 is highlighted in red)

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

**Example 2:**

Diagram: 1 → 2 → 3 → 4 → 5 → 6 (Node 4 is highlighted in red)

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.

**Constraints:**

- The number of nodes in the list is in the range `[1, 100]`.
- `1 <= Node.val <= 100`

**Code Editor:**

```
C  
1 //  
2 //  
3 struct ListNode* middleNode(struct ListNode* head) {  
4     struct ListNode* slow = head;  
5     struct ListNode* fast = head;  
6  
7     while (fast != NULL && fast->next != NULL) {  
8         slow = slow->next;  
9         fast = fast->next->next;  
10    }  
11  
12    return slow;  
13 }  
14  
15  
16  
17  
18
```

**Testcase:** **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](#) [Activate Windows](#)