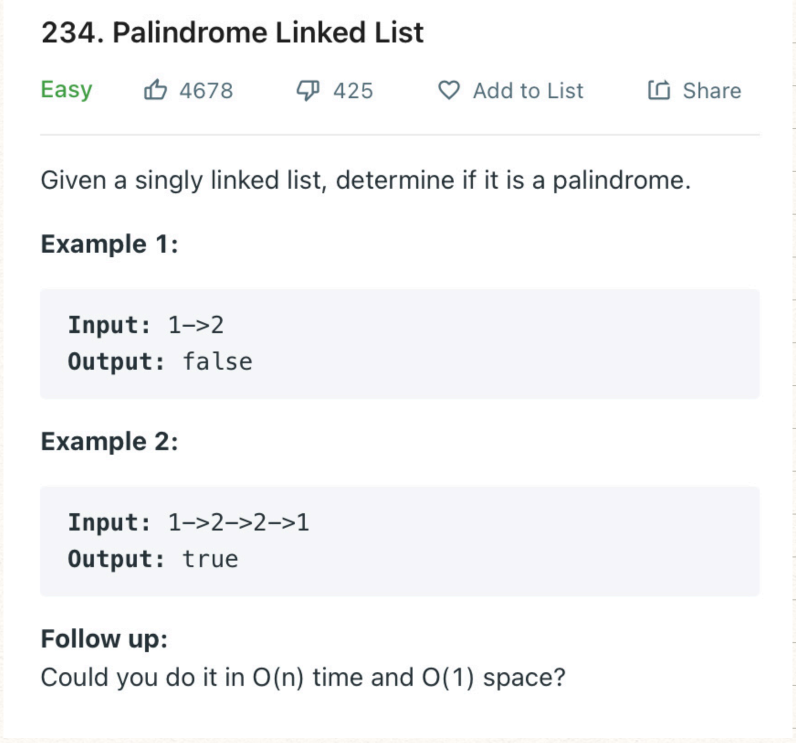


234. Palindrome Linked List

Easy 4678 425 Add to List Share

Given a singly linked list, determine if it is a palindrome.

Example 1:

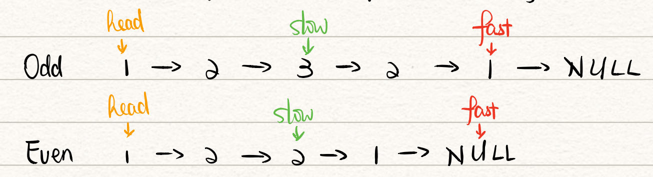
```
Input: 1->2
Output: false
```

Example 2:

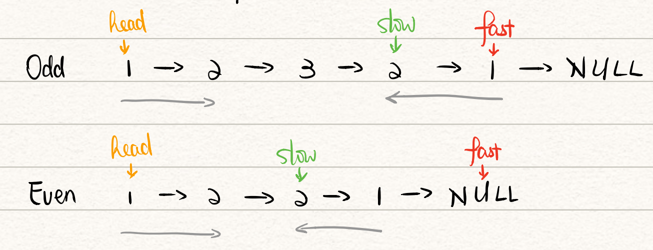
```
Input: 1->2->2->1
Output: true
```

Follow up:
Could you do it in O(n) time and O(1) space?

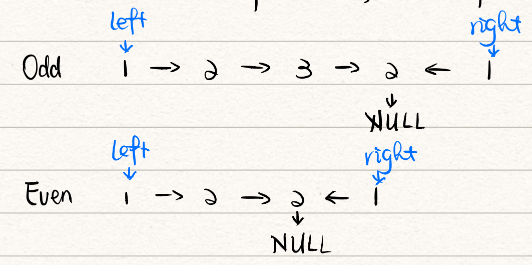
1. Use slow and fast pointers find the mid of linklist.


head
slow
fast
Odd 1 -> 2 -> 3 -> 2 -> 1 -> NULL
head
slow
fast
Even 1 -> 2 -> 2 -> 1 -> NULL

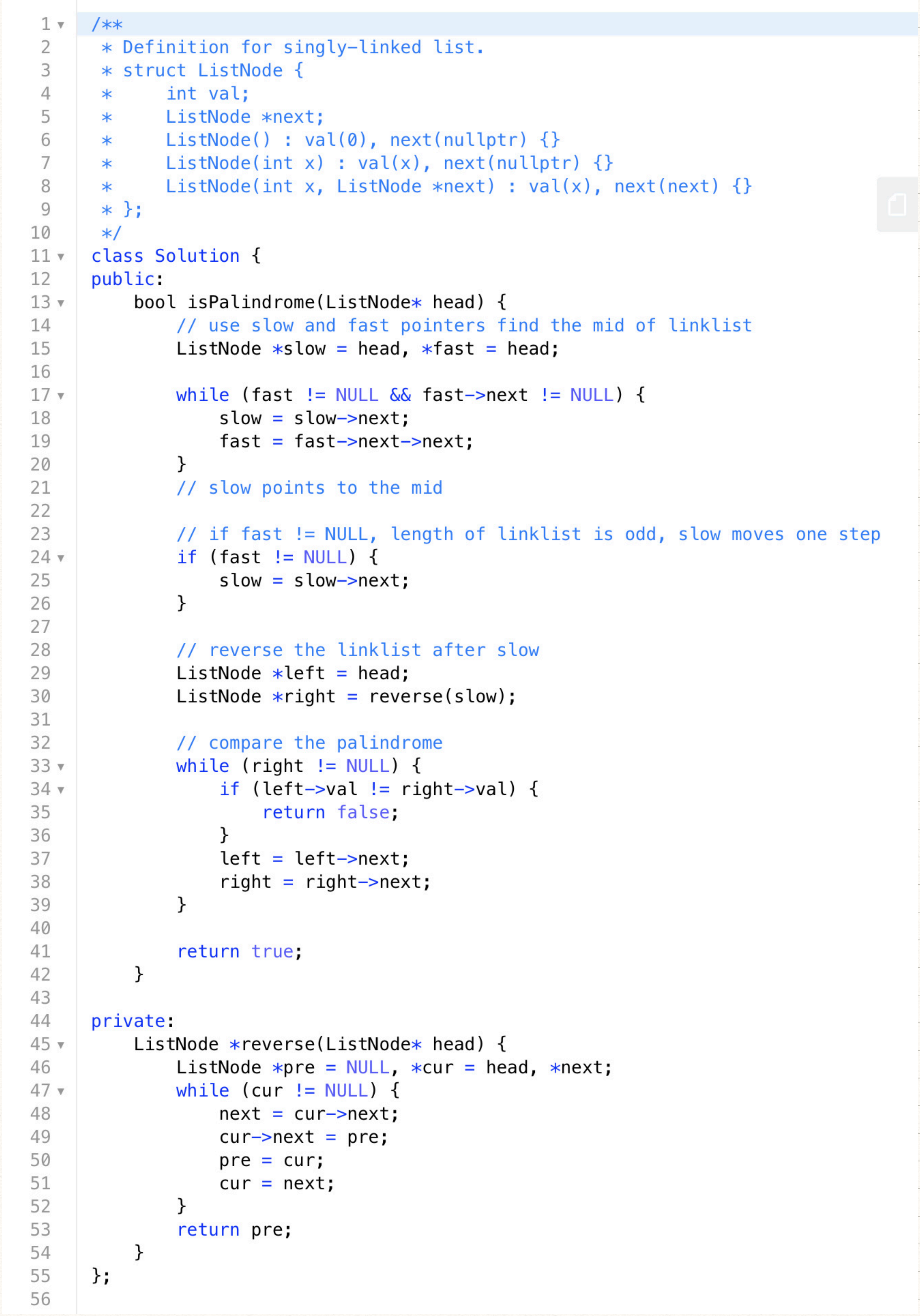
2. If fast != NULL, the length of linklist is odd, slow move one more step.


head
slow
fast
Odd 1 -> 2 -> 3 -> 2 -> 1 -> NULL
head
slow
fast
Even 1 -> 2 -> 2 -> 1 -> NULL

3. Reverse linklist after slow, and compare the palindrome.



Time complexity: O(N)
Space complexity: O(1)



```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:
    bool isPalindrome(ListNode* head) {
        // use slow and fast pointers find the mid of linklist
        ListNode *slow = head, *fast = head;

        while (fast != NULL && fast->next != NULL) {
            slow = slow->next;
            fast = fast->next->next;
        }
        // slow points to the mid

        // if fast != NULL, length of linklist is odd, slow moves one step
        if (fast != NULL) {
            slow = slow->next;
        }

        // reverse the linklist after slow
        ListNode *left = head;
        ListNode *right = reverse(slow);

        // compare the palindrome
        while (right != NULL) {
            if (left->val != right->val) {
                return false;
            }
            left = left->next;
            right = right->next;
        }

        return true;
    }

private:
    ListNode *reverse(ListNode* head) {
        ListNode *pre = NULL, *cur = head, *next;
        while (cur != NULL) {
            next = cur->next;
            cur->next = pre;
            pre = cur;
            cur = next;
        }
        return pre;
    }
};
```