Palindrome in LL in C++

```
#include <iostream>
#include <stack>

using namespace std;

// Node class definition
class Node {
public:
    int data;
    Node* next;

    // Constructor
    Node(int d) {
        data = d;
        next = nullptr;
    }
};

// LinkedList class definition
class LinkedList {
private:
    Node* head;
    Node* tail;
    int size;

public:
    // Constructor
    LinkedList() {
        head = nullptr;
        tail = nullptr;
        size = 0;
    }

    // Method to add a node at the end of the list
    void addLast(int val) {
        Node* temp = new Node(val);
        if (size == 0) {
            head = tail = temp;
        } else {
            tail->next = temp;
            tail = temp;
        }
        size++;
    }

    // Method to display the elements of the list
    void display() {
        Node* temp = head;
        while (temp != nullptr) {
            cout << temp->data << " ";
            temp = temp->next;
        }
        cout << endl;
    }

    // Method to check if the linked list is a palindrome
    bool isPalindrome() {
        Node* slow = head;
        stack<int> stack;

        // Push elements of the first half of the linked list
```

Dry Run for Your Example: **1 → 2 → 3 → 2 → 1**

Step	Stack Contents	slow points to	Comparison
Push	1, 2	3	-
Skip	(middle: 3)	2	-
Check	Top: 2 vs 2	2	✓
Check	Top: 1 vs 1	1	✓

✓ **Result: true**

Let me know if you'd like a version that modifies the list

```
onto the stack
        while (slow != nullptr) {
            stack.push(slow->data);
            slow = slow->next;
        }

        // Compare elements of the second half of the
linked list with the stack
        slow = head;
        while (slow != nullptr) {
            int top = stack.top();
            stack.pop();
            if (slow->data != top) {
                return false;
            }
            slow = slow->next;
        }

        return true;
    }
};

// Main function to demonstrate LinkedList operations
int main() {
    // Create a linked list
    LinkedList list;

    // Add elements to the linked list
    list.addLast(1);
    list.addLast(2);
    list.addLast(3);
    list.addLast(2);
    list.addLast(1);

    // Check if the linked list is a palindrome
    cout << boolalpha << list.isPalindrome() << endl; //
Output: true

    return 0;
}
```

```
true
```