Palindrome in C++

```cpp
#include <iostream>
using namespace std;

// Node class for the linked list
class Node {
public:
    int val;
    Node* next;

    Node(int val) {
        this->val = val;
        this->next = nullptr;
    }
};

// Function to find the middle node of the linked list
Node* midNode(Node* head) {
    if (head == nullptr || head->next == nullptr)
return head;

    Node* slow = head;
    Node* fast = head;

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

    return slow;
}

// Function to reverse a linked list
Node* reverseOfLL(Node* head) {
    if (head == nullptr || head->next == nullptr)
return head;

    Node* prev = nullptr;
    Node* curr = head;
    Node* forw = nullptr;

    while (curr != nullptr) {
        forw = curr->next;
        curr->next = prev;
        prev = curr;
        curr = forw;
    }

    return prev;
}

// Function to check if a linked list is a palindrome
bool isPalindrome(Node* head) {
    if (head == nullptr || head->next == nullptr)
return true;

    // Find the middle of the linked list
    Node* mid = midNode(head);

    // Reverse the second half of the list
    Node* nHead = mid->next;
```

Step-by-Step Dry Run Table

Step	Operation	Pointer/Variable	Value(s)
1	Find mid	`slow`, `fast`	Mid = 3 (slow stops here)
2	Reverse 2nd half	From node `2 -> 1`	Reversed to `1 -> 2`
3	Compare halves	`1-2-3` vs `1-2`	Matches fully
4	Restore 2nd half	Reverse back `1->2`	Back to `2->1`
5	Result		✅ `true` (Palindrome)

🧠 Output

```
true
```

```
    mid->next = nullptr;  // Split the list into two halves
    nHead = reverseOfLL(nHead);

    // Compare the two halves
    Node* c1 = head;
    Node* c2 = nHead;

    bool res = true;
    while (c2 != nullptr) {  // Only need to compare until
c2 ends
        if (c1->val != c2->val) {
            res = false;
            break;
        }
        c1 = c1->next;
        c2 = c2->next;
    }

    // Restore the original list
    nHead = reverseOfLL(nHead);
    mid->next = nHead;

    return res;
}

// Function to create a linked list from an array of
integers
Node* createList(int values[], int n) {
    Node* dummy = new Node(-1);
    Node* prev = dummy;
    for (int i = 0; i < n; ++i) {
        prev->next = new Node(values[i]);
        prev = prev->next;
    }
    return dummy->next;
}

int main() {
    // Hardcoding the linked list: 1 -> 2 -> 3 -> 2 -> 1
    int arr[] = {1, 2, 3, 2, 1};
    int n = sizeof(arr) / sizeof(arr[0]);
    Node* head = createList(arr, n);

    // Checking if the linked list is a palindrome
    cout << boolalpha << isPalindrome(head) <<
endl;  // should print true

    return 0;
}
```

Output:-

true