


Palindrome in C++																												
<pre>#include &lt;iostream&gt; using namespace std;  // Node class for the linked list class Node { public:     int val;     Node* next;      Node(int val) {         this-&gt;val = val;         this-&gt;next = nullptr;     } };  // Function to find the middle node of the linked list Node* midNode(Node* head) {     if (head == nullptr    head-&gt;next == nullptr) return head;      Node* slow = head;     Node* fast = head;      while (fast-&gt;next != nullptr &amp;&amp; fast-&gt;next-&gt;next != nullptr) {         slow = slow-&gt;next;         fast = fast-&gt;next-&gt;next;     }      return slow; }  // Function to reverse a linked list Node* reverseOfLL(Node* head) {     if (head == nullptr    head-&gt;next == nullptr) return head;      Node* prev = nullptr;     Node* curr = head;     Node* forw = nullptr;      while (curr != nullptr) {         forw = curr-&gt;next;         curr-&gt;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-&gt;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-&gt;next;</pre>		<h3>Step-by-Step Dry Run Table</h3> <table><tr><th>Step</th><th>Operation</th><th>Pointer/Variable</th><th>Value(s)</th></tr><tr><td>1</td><td>Find mid</td><td>slow, fast</td><td>Mid = 3 (slow stops here)</td></tr><tr><td>2</td><td>Reverse 2nd half</td><td>From node 2 -&gt; 1</td><td>Reversed to 1 -&gt; 2</td></tr><tr><td>3</td><td>Compare halves</td><td>1-2-3 vs 1-2</td><td>Matches fully</td></tr><tr><td>4</td><td>Restore 2nd half</td><td>Reverse back 1-&gt;2</td><td>Back to 2-&gt;1</td></tr><tr><td>5</td><td>Result</td><td></td><td>✔ true (Palindrome)</td></tr></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)
		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)																							
<div> <b>Output</b></div> <p>true</p>																												

```

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