

83. Remove duplicates from a sorted list

```
class Solution {  
public:  
    ListNode* deleteDuplicates(ListNode* head) {  
        ListNode* curr = head;  
        while (curr && curr->next) {  
            if (curr->val == curr->next->val) {  
                curr->next = curr->next->next;  
            } else {  
                curr = curr->next;  
            }  
        }  
        return head;  
    }  
};
```

The screenshot displays a code editor interface for a C++ solution. The top bar includes navigation icons and a 'Premium' badge. The left sidebar shows the 'Problem List' and 'Accepted' status. The main editor area is divided into three sections:

- Runtime and Memory:** Shows 'Accepted' status for 169/169 testcases. Runtime is 0 ms (Beats 100.00%) and Memory is 16.00 MB (Beats 99.33%). A bar chart shows the performance relative to other solutions.
- Code:** Displays the C++ code for the solution, which uses a while loop to traverse the list and skip duplicate nodes.
- Testcase and Test Result:** Shows the results for 'Case 1' with input [1,1,2] and output [1,2]. The status is 'Accepted' with a runtime of 0 ms.

The code in the editor is as follows:

```
//  
// 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:  
    ListNode* deleteDuplicates(ListNode* head) {  
        ListNode* curr = head;  
        while (curr && curr->next) {  
            if (curr->val == curr->next->val) {  
                curr->next = curr->next->next;  
            } else {  
                curr = curr->next;  
            }  
        }  
        return head;  
    }  
};
```