TIPS template

# Interviewer:

## Behavioral:

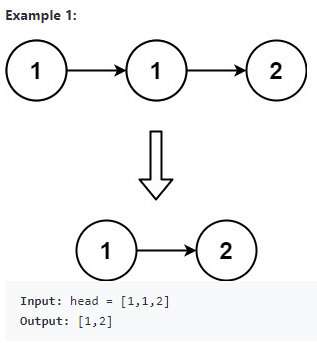
What company are you most interested in applying to? Please give me your elevator pitch/introduction that you would give to that company.

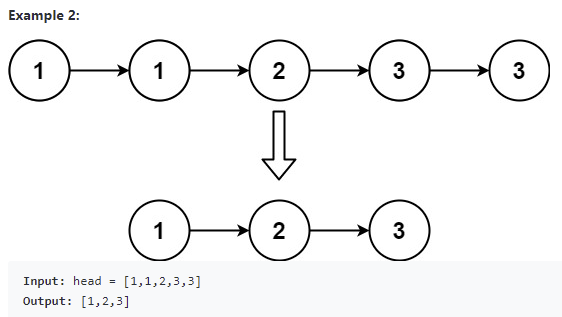
## Question:

<https://leetcode.com/problems/remove-duplicates-from-sorted-list/>

83. Remove Duplicates from Sorted List  
Given the head of a sorted linked list, *delete all duplicates such that each element appears only once*. Return *the linked list sorted as well*

## Examples:

**

**

## Follow up Q&A:

* Will the head be null?
  + Possibly
* Is the linked list singly or doubly linked?
  + It is singly linked.
* What is the optimal time/space complexity for this problem?
  + What do you think?
    - Hint: the answer is O(n)

## Hint(s):

*Ask if they would like a hint before giving a hint*

1. Remember that the list is sorted
2. You should be able to do it in one pass because it is sorted.
3. Duplicates should be next to each other in a sorted list.

## Solution(s): (General concept and time/space complexity)

#### Approach 1: Straight-Forward Approach

This is a simple problem that merely tests your ability to manipulate list node pointers. Because the input list is sorted, we can determine if a node is a duplicate by comparing its value to the node *after* it in the list. If it is a duplicate, we change the next pointer of the current node so that it skips the next node and points directly to the one after the next node.

public ListNode deleteDuplicates(ListNode head) {

ListNode current = head;

while (current != null && current.next != null) {

if (current.next.val == current.val) {

current.next = current.next.next;

} else {

current = current.next;

}

}

return head;

}

Time complexity: O(n)

Space complexity: O(1)

### Other questions follow up

*Ask if there is more than 5 minutes remaining when they finish their code and testing.*

* How would you change your solution if we only allowed 2 nodes of a single number?

# Interviewee:

## Question:

<https://leetcode.com/problems/valid-palindrome/>

Given a string s, determine if it is a palindrome, considering only alphanumeric characters and ignoring cases.

## Example(s):

Input: s = "race a car"

Output: false

Explanation: "raceacar" is not a palindrome.

Input: s = "A man, a plan, a canal: Panama"

Output: true

Explanation: "amanaplanacanalpanama" is a palindrome.

## Code below or on leetcode