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/valid-palindrome/>

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

## Examples:

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.

## Follow up Q&A:

## Hint(s):

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

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

bool isPalindrome(string s) {

for (int i = 0, j = s.size() - 1; i < j; i++, j--) { // Move 2 pointers from each end until they collide

while (isalnum(s[i]) == false && i < j) i++; // Increment left pointer if not alphanumeric

while (isalnum(s[j]) == false && i < j) j--; // Decrement right pointer if no alphanumeric

if (toupper(s[i]) != toupper(s[j])) return false; // Exit and return error if not match

}

return true;

}

### Name of Solution

If you take any ordinary string, and concatenate its reverse to it, you'll get a palindrome. This leads to an interesting insight about the converse: every palindrome half is reverse of the other half.

Simply speaking, if one were to start in the middle of a palindrome, and traverse outwards, they'd encounter the same characters, in the exact same order, in both halves!

Since the input string contains characters that we need to ignore in our palindromic check, it becomes tedious to figure out the real middle point of our palindromic input.

Instead of going outwards from the middle, we could just go inwards towards the middle!

So, if we start traversing inwards, from both ends of the input string, we can expect to *see* the same characters, in the same order.

The resulting algorithm is simple:

* Set two pointers, one at each end of the input string
* If the input is palindromic, both the pointers should point to equivalent characters, *at all times*. [[1]](https://leetcode.com/problems/valid-palindrome/solution/#fn1)
  + If this condition is not met at any point of time, we break and return early. [[2]](https://leetcode.com/problems/valid-palindrome/solution/#fn2)
* We can simply ignore non-alphanumeric characters by continuing to traverse further.
* Continue traversing inwards until the pointers meet in the middle.
* Time complexity :
* O(n)
* *O*(*n*), in length
* n
* *n* of the string. We traverse over each character at-most once, until the two pointers meet in the middle, or when we break and return early.
* Space complexity :
* O(1)
* *O*(1). No extra space required, at all.

### Other questions follow up

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

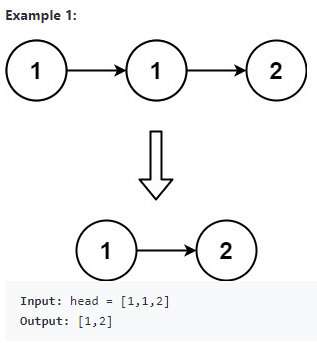
# Interviewee:

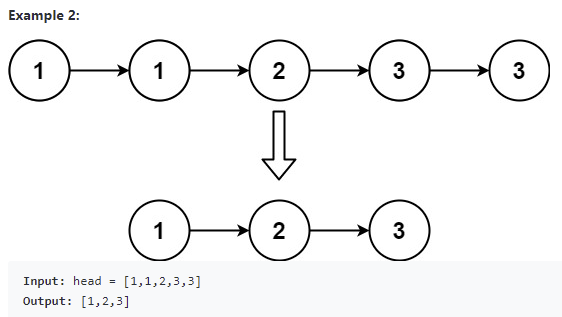
## Question:

<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*

## Example(s):

**

**

## 

## Code below or on leetcode