1. **[Two Sum](https://leetcode.com/problems/two-sum/)**

# **Solution:**

var twoSum = function(nums, target) {

    // Outer loop: Iterate over the array, considering each element as the first number.

    for (let i = 0; i < nums.length; i++) {

        // Inner loop: Iterate over the rest of the array starting from the element after the current element of the outer loop.

        for (let j = i + 1; j < nums.length; j++) {

            // Check if the sum of the current pair of numbers equals the target.

            if (nums[i] + nums[j] == target) {

                // If found, return the indices of the two numbers that sum up to the target.

                return [i, j];

            }

        }

    }

    // If no valid pair is found, return an empty array.

    return [];

};

[**58. Length of Last Word**](https://leetcode.com/problems/length-of-last-word/)

# **Solution:**

var lengthOfLastWord = function(s) {

    // Step 1: Remove any leading or trailing spaces from the string

    s = s.trim();

    // Step 2: Initialize a variable to keep track of the length of the last word

    let length = 0;

    // Step 3: Loop through the string in reverse order, starting from the end

    for (let i = s.length - 1; i >= 0; i--) {

        // If the current character is not a space, it's part of the last word

        if (s[i] !== ' ') {

            length++;

        }

        // If a space is encountered after counting characters, break the loop

        else if (length > 0) {

            break;

        }

    }

    // Step 4: Return the length of the last word

    return length;

};

[**747. Largest Number At Least Twice of Others**](https://leetcode.com/problems/largest-number-at-least-twice-of-others/)

# **Solution:**

var dominantIndex = function(nums) {

let max = Math.max(...nums); // Find the largest number

  let maxIndex = nums.indexOf(max); // Find the index of the largest number

  // Check if the largest number is at least twice as much as every other number

  for (let i = 0; i < nums.length; i++) {

    if (i !== maxIndex && max < nums[i] \* 2) {

      return -1; // If any number is greater than half of max, return -1

    }

  }

  return maxIndex; // Return the index of the largest element if condition is met

};

[**125. Valid Palindrome**](https://leetcode.com/problems/valid-palindrome/)

# **Solution:**

var isPalindrome = function(s) {

     // Step 1: Convert all characters to lowercase and remove non-alphanumeric characters

  const cleanedString = s.toLowerCase().replace(/[^a-z0-9]/g, '');

  // Step 2: Use two pointers to check if the string is the same forward and backward

  let left = 0;

  let right = cleanedString.length - 1;

  while (left < right) {

    if (cleanedString[left] !== cleanedString[right]) {

      return false; // If characters at left and right don't match, it's not a palindrome

    }

    left++;

    right--;

  }

  return true; // If all characters match, it's a palindrome

};

[**997. Find the Town Judge**](https://leetcode.com/problems/find-the-town-judge/)  
**solution:**

var findJudge = function (n, trust) {

  // Step 1: Initialize arrays for trust count and out degree

  const trustCount = new Array(n).fill(0);

  const outDegree = new Array(n).fill(0);

  // Step 2: Process each trust relationship

  for (const [a, b] of trust) {

    outDegree[a - 1]++; // Person 'a' trusts person 'b'

    trustCount[b - 1]++; // Person 'b' is trusted by person 'a'

  }

  // Step 3: Find the town judge

  for (let i = 0; i < n; i++) {

    if (outDegree[i] === 0 && trustCount[i] === n - 1) {

      return i + 1; // Return 1-based index

    }

  }

  // Step 4: If no town judge found

  return -1;

}

**[383. Ransom Note](https://leetcode.com/problems/ransom-note/)**

# **Solution:**

var canConstruct = function (ransomNote, magazine) {

  const count = new Array(26).fill(0); // Array to store frequency of letters in magazine

  // Count frequency of each letter in magazine

  for (let i = 0; i < magazine.length; i++) {

    count[magazine.charCodeAt(i) - 'a'.charCodeAt(0)]++;

  }

  // Check if ransomNote can be constructed from magazine

  for (let i = 0; i < ransomNote.length; i++) {

    const index = ransomNote.charCodeAt(i) - 'a'.charCodeAt(0);

    if (count[index] > 0) {

      count[index]--; // Decrease the count for this character in magazine

    } else {

      return false; // Not enough letters in magazine

    }

  }

  return true; // All characters in ransomNote can be constructed

}