❓ solution ❓

1️⃣ A word is on the loose and now has tried to hide amongst a crowd of tall letters, help write a function to detect what the word is, knowing the following rules:

⚠️ the word of interest is in lowercase

⚠️ the crowd of letters is all in uppercase

⚠️ note the word will be spread out amongst the random letters, but their letters remain in the same order.

```
function collectLowercase(str) {
    let lowercase = '';
        for (let char of str) {
        if (char >= 'a' && char <= 'z') {
            lowercase += char;
        }
    }

    return lowercase;
}
```

2️⃣ Create a function that returns true if the first array can be nested inside the second array.

Example: passed argument [3,4,5] and [2,5,7,8] answer: return true

```
function canBeNested(arr1, arr2) {
    // Sort both arrays
    arr1.sort((a, b) => a - b);
    arr2.sort((a, b) => a - b);

    // Check if the smallest element of arr1 is greater than the smallest element
of arr2
    // and if the largest element of arr1 is smaller than the largest element of
arr2
    return arr1[0] > arr2[0] && arr1[arr1.length - 1] < arr2[arr2.length - 1];
}

// Example usage:
const array1 = [1, 2, 3];
const array2 = [0, 1, 2, 3, 4];
console.log(canBeNested(array1, array2));
```

3️⃣ Magic array exercise

an array is defined to be a magic array if the sum of the prime in the array is equal to the first element of the array . if there are no primes in the array ,the first element must be 0. so{21,3,7,9,11,4,6} is a magic array because 3,7,11are the prime in the array and they sum to 21 which is the first element of the array.{13,4,4,4,4} is also a magic array because the sum of the prime is 13 which is also the first element.other magic array are {10,5,5},but {0,6,8,20} and{3},{8,5,-5,5,3} is not a magic array because the sum of the prims is 5+5+5 = 13.

Note that -5 is not a prime because prime numbers are positive.

```
function isMagicArray(arr) {
    // Helper function to check if a number is prime
    function isPrime(num) {
        if (num <= 1) return false;
        for (let i = 2; i <= Math.sqrt(num); i++) {
            if (num % i === 0) return false;
        }
        return true;
    }

    // Find the sum of prime numbers in the array
    let sum = 0;
    for (let i = 0; i < arr.length; i++) {
        if (isPrime(arr[i])) {
            sum += arr[i];
        }
    }

    // Check if the sum of primes equals the first element of the array, or if
there are no primes and the first element is 0
    return (sum === arr[0] && sum !== 0) || (sum === 0 && arr[0] === 0);
}

// Example usage:
console.log(isMagicArray([21, 3, 7, 9, 11, 4, 6])); // Output: true
console.log(isMagicArray([13, 4, 4, 4, 4])); // Output: true
console.log(isMagicArray([10, 5, 5])); // Output: true
console.log(isMagicArray([0, 6, 8, 20])); // Output: false
console.log(isMagicArray([3])); // Output: false
console.log(isMagicArray([8, 5, -5, 5, 3])); // Output: false
```

4️⃣ Create a function that takes an array of numbers and returns both the minimum and maximum numbers, in that order inside another array.

▶️ **Example : passed argument [1,2,3,4,5] answer : return [1,5]**

```
function findMinMax(nums) {
    if (nums.length === 0) {
        return [];
    }

    let min = nums[0];
    let max = nums[0];

    for (let i = 1; i < nums.length; i++) {
        if (nums[i] < min) {
            min = nums[i];
        }
        if (nums[i] > max) {
            max = nums[i];
        }
    }

    return [min, max];
}

// Example usage:
console.log(findMinMax([1, 2, 3, 4, 5])); // Output: [1, 5]
console.log(findMinMax([10, -5, 7, 3, -2])); // Output: [-5, 10]
console.log(findMinMax([])); // Output: []
```

5️⃣ Create a function that takes a number as its argument and returns an array of all its factors.

▶️ **Example**: passed argument 12 answer: return [1,2,3,4,6,12]

```
function findFactors(num) {
    const factors = [];

    for (let i = 1; i <= num; i++) {
        if (num % i === 0) {
            factors.push(i);
        }
    }

    return factors;
}

// Example usage:
console.log(findFactors(12)); // Output: [1, 2, 3, 4, 6, 12]
console.log(findFactors(16)); // Output: [1, 2, 4, 8, 16]
console.log(findFactors(7)); // Output: [1, 7]
```

6️⃣ Given a number, return an array containing the two halves of the number. If the number is odd, make the rightmost number higher.

▶️ **Example** : passed argument 4 answer: return [2,2]

```
function splitNumber(num) {
    const half = Math.floor(num / 2);
    const leftHalf = half;
    const rightHalf = num - half;

    return [leftHalf, rightHalf];
}

// Example usage:
console.log(splitNumber(10)); // Output: [5, 5]
console.log(splitNumber(11)); // Output: [5, 6]
console.log(splitNumber(7)); // Output: [3, 4]
```