Create a function that takes two numbers as arguments and return their sum.

addition(3, 2) ➞ 5

addition(-3, -6) ➞ -9

addition(7, 3) ➞ 10

Write a function that takes an integer minutes and converts it to seconds.

convert(5) ➞ 300

convert(3) ➞ 180

convert(2) ➞ 120

Create a function that takes a number as an argument, increments the number by +1 and returns the result.

addition(0) ➞ 1

addition(9) ➞ 10

addition(-3) ➞ -2

Write a function that takes the base and height of a triangle and return its area.

triArea(3, 2) ➞ 3

triArea(7, 4) ➞ 14

triArea(10, 10) ➞ 50

* The area of a triangle is: (base \* height) / 2

Write a function that returns the string "something" joined with a space " " and the given argument a.

giveMeSomething("is better than nothing") ➞ "something is better than nothing"

giveMeSomething("Bob Jane") ➞ "something Bob Jane"

giveMeSomething("something") ➞ "something something"

Write a function that converts hours into seconds.

howManySeconds(2) ➞ 7200

howManySeconds(10) ➞ 36000

howManySeconds(24) ➞ 86400

* 60 seconds in a minute, 60 minutes in an hour

Create a function that takes an array and returns the first element.

getFirstValue([1, 2, 3]) ➞ 1

getFirstValue([80, 5, 100]) ➞ 80

getFirstValue([-500, 0, 50]) ➞ -500

Create a function that finds the maximum range of a triangle's third edge, where the side lengths are all integers.

nextEdge(8, 10) ➞ 17

nextEdge(5, 7) ➞ 11

nextEdge(9, 2) ➞ 10

* (side1 + side2) - 1 = maximum range of third edge.
* The side lengths of the triangle are positive integers.
* Don't forget to return the result.

**Return the Remainder from Two Numbers.** There is a single operator in JavaScript, capable of providing the remainder of a division operation. Two numbers are passed as parameters. The first parameter divided by the second parameter will have a remainder, possibly zero. Return that value.

remainder(1, 3) ➞ 1

remainder(3, 4) ➞ 3

remainder(-9, 45) ➞ -9

remainder(5, 5) ➞ 0

* The tests only use positive and negative integers.
* Don't forget to return the result.

Create a function that takes length and width and finds the perimeter of a rectangle.

findPerimeter(6, 7) ➞ 26

findPerimeter(20, 10) ➞ 60

findPerimeter(2, 9) ➞ 22

* Don't forget to return the result.

Create a function that takes a number as its only argument and returns true if it's less than or equal to zero, otherwise return false.

lessThanOrEqualToZero(5) ➞ false

lessThanOrEqualToZero(0) ➞ true

lessThanOrEqualToZero(-2) ➞ true

In this challenge, you must verify the equality of two different values given the parameters a and b.

Both the *value* and *type* of the parameters need to be equal. The possible types of the given parameters are:

* Numbers
* Strings
* Booleans (false or true)
* Special values: undefined, null and NaN

What have you learned so far that will permit you to do two different checks (value **and** type) with a single statement?

Implement a function that returns true if the parameters are equal, and false if they are not.

checkEquality(1, true) ➞ false

// A number and a boolean: the value and type are different.

checkEquality(0, "0") ➞ false

// A number and a string: the type is different.

checkEquality(1, 1) ➞ true

// A number and a number: the type and value are equal.

Create a function that returns true if a string is empty and false otherwise.

isEmpty("") ➞ true

isEmpty(" ") ➞ false

isEmpty("a") ➞ false

* A string containing only whitespaces " " does **not** count as empty.
* Don't forget to return the result.

Create a function that takes two strings as arguments and return either true or false depending on whether the total number of characters in the first string is equal to the total number of characters in the second string.

comp("AB", "CD") ➞ true

comp("ABC", "DE") ➞ false

comp("hello", "edabit") ➞ false

Write a function that returns the length of a string. **Make your function recursive.**

length("apple") ➞ 5

length("make") ➞ 4

length("a") ➞ 1

length("") ➞ 0

Create a function that returns true if an integer is evenly divisible by 5, and false otherwise.

divisibleByFive(5) ➞ true

divisibleByFive(-55) ➞ true

divisibleByFive(37) ➞ false

Create a function that takes an integer and return true if it's divisible by 100, otherwise return false.

divisible(1) ➞ false

divisible(1000) ➞ true

divisible(100) ➞ true

Create a function that takes a string and returns it as an integer.

stringInt("6") ➞ 6

stringInt("1000") ➞ 1000

stringInt("12") ➞ 12

Given two strings, firstName and lastName, return a single string in the format "last, first".

concatName("First", "Last") ➞ "Last, First"

concatName("John", "Doe") ➞ "Doe, John"

concatName("Mary", "Jane") ➞ "Jane, Mary"

Write a function that checks whether a person can watch an MA15+ rated movie. One of the following two conditions is required for admittance:

* The person is at least 15 years old.
* They have parental supervision.

The function accepts two parameters, age and isSupervised. Return a boolean.

acceptIntoMovie(14, true) ➞ true

acceptIntoMovie(14, false) ➞ false

acceptIntoMovie(16, false) ➞ true

* age is a decimal.
* isSupervised is a boolean.

Write a function to reverse an array.

reverse([1, 2, 3, 4]) ➞ [4, 3, 2, 1]

reverse([9, 9, 2, 3, 4]) ➞ [4, 3, 2, 9, 9]

reverse([]) ➞ []

Create a function that takes an array of numbers or letters and returns a string.

arrayToString([1, 2, 3, 4, 5, 6]) ➞ "123456"

arrayToString(["a", "b", "c", "d", "e", "f"]) ➞ "abcdef"

arrayToString([1, 2, 3, "a", "s", "dAAAA"]) ➞ "123asdAAAA"

Create a function that accepts an array and returns the last item in the array.

getLastItem([1, 2, 3]) ➞ 3

getLastItem(["cat", "dog", "duck"]) ➞ "duck"

getLastItem([true, false, true]) ➞ true

Create a function that takes a word and returns the new word without including the first character.

newWord("apple") ➞ "pple"

newWord("cherry") ➞ "herry"

newWord("plum") ➞ "lum"

Create a function that takes an array and a string as arguments and return the index of the string.

findIndex(["hi", "edabit", "fgh", "abc"], "fgh") ➞ 2

findIndex(["Red", "blue", "Blue", "Green"], "blue") ➞ 1

findIndex(["a", "g", "y", "d"], "d") ➞ 3

findIndex(["Pineapple", "Orange", "Grape", "Apple"], "Pineapple") ➞ 0