Algorithm challenges

BASIC CHALLENGES

Reverse a String

Reverse the provided string. You may need to turn the string into an array before you can reverse it. Your result must be a string.

- `reverseString("hello")` should return a string.
- `reverseString("hello")` should become`"olleh"`.
- `reverseString("Howdy")` should become`"ydwoH"`.
- `reverseString("Greetings from Earth")`should return `"htraE morf sgniteerG"`.

Factorialize a number

If the integer is represented with the letter n, a factorial is the product of all positive integers less than or equal to n.

Factorials are often represented with the shorthand notation `n!`

For example: `5! = 1 * 2 * 3 * 4 * 5 = 120`

- `factorialize(5)` should return a number.
- `factorialize(5)` should return 120.
- `factorialize(10)` should return 3628800.
- `factorialize(20)` should return 2432902008176640000.
- `factorialize(0)` should return 1.

Check for palindromes

Return `true` if the given string is a palindrome. Otherwise, return `false`.

A `palindrome` is a word or sentence that's spelled the same way both forward and backward, ignoring punctuation, case, and spacing.

Note

You'll need to remove **all non-alphanumeric characters**(punctuation, spaces and symbols) and turn everything lower case in order to check for palindromes.

We'll pass strings with varying formats, such as`"racecar"`, `"RaceCar"`, and `"race CAR"` among others.

We'll also pass strings with special symbols, such as`"2A3*3a2"`, `"2A3 3a2"`, and `"2_A3*3#A2"`.

- `palindrome("eye")` should return a boolean.
- `palindrome("eye")` should return true.
- `palindrome("_eye")` should return true.
- `palindrome("race car")` should return true.
- `palindrome("not a palindrome")` should return false.
- `palindrome("A man, a plan, a canal. Panama")` should return true.
- `palindrome("never odd or even")` should return true.
- `palindrome("nope")` should return false.
- `palindrome("almostomla")` should return false.
- `palindrome("My age is 0, 0 si ega ym.")`should return true.
- `palindrome("1 eye for of 1 eye.")`should return false.
- `palindrome("0_0 (: /-\ :) 0-0")` should return true.
- `palindrome("five|\_/|four")` should return false.

Find the Longest Word in a String

Return the length of the longest word in the provided sentence.

Your response should be a number.

- `findLongestWord("The quick brown fox jumped over the lazy dog")` should return a number.
- `findLongestWord("The quick brown fox jumped over the lazy dog")` should return 6.
- `findLongestWord("May the force be with you")` should return 5.
- `findLongestWord("Google do a barrel roll")` should return 6.
- `findLongestWord("What is the average airspeed velocity of an unladen swallow")` should return 8.
- `findLongestWord("What if we try a super-long word such as otorhinolaryngology")`should return 19.

Title Case a Sentence

Return the provided string with the first letter of each word capitalized. Make sure the rest of the word is in lower case.

For the purpose of this exercise, you should also capitalize connecting words like "the" and "of".

- `titleCase("I'm a little tea pot")`should return a string.
- `titleCase("I'm a little tea pot")`should return "I'm A Little Tea Pot".
- `titleCase("sHoRt AnD sToUt")` should return "Short And Stout".
- `titleCase("HERE IS MY HANDLE HERE IS MY SPOUT")` should return "Here Is My Handle Here Is My Spout".

Return Largest Numbers in Arrays

Return an array consisting of the largest number from each provided sub-array. For simplicity, the provided array will contain exactly 4 sub-arrays.

Remember, you can iterate through an array with a simple for loop, and access each member with array syntax `arr[i]`.

- `largestOfFour([[4, 5, 1, 3], [13, 27, 18, 26], [32, 35, 37, 39], [1000, 1001, 857, 1]])` should return an array.
- `largestOfFour([[13, 27, 18, 26], [4, 5, 1, 3], [32, 35, 37, 39], [1000, 1001, 857, 1]])` should return`[27,5,39,1001]`.
- `largestOfFour([[4, 9, 1, 3], [13, 35, 18, 26], [32, 35, 97, 39], [1000000, 1001, 857, 1]])` should return `[9, 35, 97, 1000000]`.

Confirm the Ending

Check if a string (first argument, `str`) ends with the given target string (second argument, `target`).

This challenge *can* be solved with the `.endsWith()`method, which was introduced in ES2015. But for the purpose of this challenge, we would like you to use one of the JavaScript substring methods instead.

- `confirmEnding("Bastian", "n")` should return true.
- `confirmEnding("Connor", "n")` should return false.
- `confirmEnding("Walking on water and developing software from a specification are easy if both are frozen", "specification")` should return false.
- `confirmEnding("He has to give me a new name", "name")` should return true.
- `confirmEnding("Open sesame", "same")`should return true.
- `confirmEnding("Open sesame", "pen")`should return false.

- `confirmEnding("If you want to save our world, you must hurry. We dont know how much longer we can withstand the nothing", "mountain")` should return false.
- Do not use the built-in method `.endsWith()` to solve the challenge.

Repeat a string repeat a string

Repeat a given string (first argument) `num` times (second argument). Return an empty string if `num` is not a positive number.

- `repeatStringNumTimes("*", 3)` should return `"***"`.
- `repeatStringNumTimes("abc", 3)` should return `"abcabcabc"`.
- `repeatStringNumTimes("abc", 4)` should return `"abcabcabcabc"`.
- `repeatStringNumTimes("abc", 1)` should return `"abc"`.
- `repeatStringNumTimes("*", 8)` should return `"********"`.
- `repeatStringNumTimes("abc", -2)` should return `""`.

Truncate a string

Truncate a string (first argument) if it is longer than the given maximum string length (second argument). Return the truncated string with a `...` ending.

Note that inserting the three dots to the end will add to the string length.

However, if the given maximum string length `num` is less than or equal to 3, then the addition of the three dots does not add to the string length in determining the truncated string.

- `truncateString("A-tisket a-tasket A green and yellow basket", 11)` should return "A-tisket...".
- `truncateString("Peter Piper picked a peck of pickled peppers", 14)` should return "Peter Piper...".
- `truncateString("A-tisket a-tasket A green and yellow basket", "A-tisket a-tasket A green and yellow basket".length)` should return "A-tisket a-tasket A green and yellow basket".

- `truncateString("A-tisket a-tasket A green and yellow basket", "A-tisket a-tasket A green and yellow basket".length + 2)` should return "A-tisket a-tasket A green and yellow basket".
- `truncateString("A-", 1)` should return "A...".
- `truncateString("Absolutely Longer", 2)`should return "Ab...".

Chunky Monkey

Write a function that splits an array (first argument) into groups the length of `size` (second argument) and returns them as a two-dimensional array.

- `chunkArrayInGroups(["a", "b", "c", "d"], 2)` should return `[["a", "b"], ["c", "d"]]`.
- `chunkArrayInGroups([0, 1, 2, 3, 4, 5], 3)` should return `[[0, 1, 2], [3, 4, 5]]`.
- `chunkArrayInGroups([0, 1, 2, 3, 4, 5], 2)` should return `[[0, 1], [2, 3], [4, 5]]`.
- `chunkArrayInGroups([0, 1, 2, 3, 4, 5], 4)` should return `[[0, 1, 2, 3], [4, 5]]`.
- `chunkArrayInGroups([0, 1, 2, 3, 4, 5, 6], 3)` should return `[[0, 1, 2], [3, 4, 5], [6]]`.
- `chunkArrayInGroups([0, 1, 2, 3, 4, 5, 6, 7, 8], 4)` should return `[[0, 1, 2, 3], [4, 5, 6, 7], [8]]`.
- `chunkArrayInGroups([0, 1, 2, 3, 4, 5, 6, 7, 8], 2)` should return `[[0, 1], [2, 3], [4, 5], [6, 7], [8]]`.

Slasher Flick

Return the remaining elements of an array after chopping off `n` elements from the head.

The head means the beginning of the array, or the zeroth index.

- `slasher([1, 2, 3], 2)` should return`[3]`.
- `slasher([1, 2, 3], 0)` should return`[1, 2, 3]`.
- `slasher([1, 2, 3], 9)` should return`[]`.

- `slasher([1, 2, 3], 4)` should return`[]`.
- `slasher(["burgers", "fries", "shake"], 1)` should return `["fries", "shake"]`.
- `slasher([1, 2, "chicken", 3, "potatoes", "cheese", 4], 5)` should return`["cheese", 4]`.

Falsy Bouncer

Remove all falsy values from an array.

Falsy values in JavaScript are `false`, `null`, `0`, `""`,`undefined`, and `NaN`.

- `bouncer([7, "ate", "", false, 9])`should return `[7, "ate", 9]`.
- `bouncer(["a", "b", "c"])` should return`["a", "b", "c"]`.
- `bouncer([false, null, 0, NaN, undefined, ""])` should return `[]`.
- `bouncer([1, null, NaN, 2, undefined])`should return `[1, 2]`.

Seek and Destroy

You will be provided with an initial array (the first argument in the destroyer function), followed by one or more arguments. Remove all elements from the initial array that are of the same value as these arguments.

- `destroyer([1, 2, 3, 1, 2, 3], 2, 3)`should return `[1, 1]`.
- `destroyer([1, 2, 3, 5, 1, 2, 3], 2, 3)`should return `[1, 5, 1]`.
- `destroyer([3, 5, 1, 2, 2], 2, 3, 5)`should return `[1]`.
- `destroyer([2, 3, 2, 3], 2, 3)` should return `[]`.
- `destroyer(["tree", "hamburger", 53], "tree", 53)` should return`["hamburger"]`.

Where do I belong

Return the lowest index at which a value (second argument) should be inserted into an array (first argument) once it has been sorted. The returned value should be a number.

For example, `getIndexToIns([1,2,3,4], 1.5)` should return `1` because it is greater than `1` (index 0), but less than `2` (index 1).

Likewise, `getIndexToIns([20,3,5], 19)` should return`2` because once the array has been sorted it will look like`[3,5,20]` and `19` is less than `20` (index 2) and greater than `5` (index 1).

- `getIndexToIns([10, 20, 30, 40, 50], 35)`should return `3`.
- `getIndexToIns([10, 20, 30, 40, 50], 30)`should return `2`.
- `getIndexToIns([40, 60], 50)` should return `1`.
- `getIndexToIns([3, 10, 5], 3)` should return `0`.
- `getIndexToIns([5, 3, 20, 3], 5)` should return `2`.
- `getIndexToIns([2, 20, 10], 19)` should return `2`.
- `getIndexToIns([2, 5, 10], 15)` should return `3`.

Caesars Cipher

One of the simplest and most widely known `ciphers` is a`Caesar cipher`, also known as a `shift cipher`. In a`shift cipher` the meanings of the letters are shifted by some set amount.

A common modern use is the **ROT13** cipher, where the values of the letters are shifted by 13 places. Thus 'A' ↔ 'N', 'B' ↔ 'O' and so on.

Write a function which takes a **ROT13** encoded string as input and returns a decoded string.

All letters will be uppercase. Do not transform any non-alphabetic character (i.e. spaces, punctuation), but do pass them on.

- `rot13("SERR PBQR PNZC")` should decode to `"FREE CODE CAMP"`
- `rot13("SERR CVMMN!")` should decode to`"FREE PIZZA!"`
- `rot13("SERR YBIR?")` should decode to`"FREE LOVE?"`
- `rot13("GUR DHVPX OEBJA QBT WHZCRQ BIRE GUR YNML SBK.")` should decode to `"THE QUICK BROWN DOG JUMPED OVER THE LAZY FOX."`