Secuencias de comandos de algoritmos básicos

**Convertir Celsius a Fahrenheit**

La fórmula para convertir de Celsius a Fahrenheit es la temperatura en Celsius veces 9/5, más 32.

Se le da una variable celsiusque representa una temperatura en Celsius. Utilice la variable fahrenheitya definida y asígnele la temperatura Fahrenheit equivalente a la temperatura Celsius dada. Use la fórmula mencionada anteriormente para ayudar a convertir la temperatura Celsius a Fahrenheit.

**function convertCtoF(celsius) {**

**let fahrenheit = 9/5 \* celsius + 32;**

**return fahrenheit;**

**}**

**convertCtoF(30);**

**Invertir una cadena**

Invierta la cadena proporcionada y devuelva la cadena invertida.

Por ejemplo, "hello"debería convertirse en "olleh".

**function reverseString(str) {**

**let arr =str.split('').reverse();**

**console.log(arr)**

**return arr.join('');**

**}**

**reverseString("hello");**

**Factorizar un número**

Devuelve el factorial del entero proporcionado.

Si el entero se representa con la letra n, un factorial es el producto de todos los enteros positivos menores o iguales a n.

Los factoriales a menudo se representan con la notación abreviadan!

Por ejemplo:5! = 1 \* 2 \* 3 \* 4 \* 5 = 120

Solo se proporcionarán a la función números enteros mayores o iguales a cero.

**function factorialize(num) {**

**let product = 1;**

**for (let i = 2; i <= num; i++) {**

**product \*= i;**

**}**

**return product;**

**}**

**factorialize(5);**

**Encuentra la palabra más larga en una cadena**

Devuelve la longitud de la palabra más larga en la oración proporcionada.

Su respuesta debe ser un número.

**function findLongestWordLength(str) {**

**let words = str.split(' ');**

**let maxLength = 0;**

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

**if (words[i].length > maxLength) {**

**maxLength = words[i].length;**

**}**

**}**

**return maxLength;**

**}**

**findLongestWordLength("The quick brown fox jumped over the lazy dog");**

**Devolver los números más grandes en matrices**

Devuelve una matriz que consiste en el número más grande de cada subarreglo proporcionado. Para simplificar, la matriz proporcionada contendrá exactamente 4 sub-matrices.

Recuerde, puede iterar a través de una matriz con un bucle for simple y acceder a cada miembro con sintaxis de matriz arr[i].

**function largestOfFour(arr) {**

**const results = [];**

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

**let largestNumber = arr[i][0];**

**for (let j = 1; j < arr[i].length; j++) {**

**if (arr[i][j] > largestNumber) {**

**largestNumber = arr[i][j];**

**}**

**}**

**results[i] = largestNumber;**

**}**

**return results;**

**}**

**largestOfFour([[4, 5, 1, 3], [13, 27, 18, 26], [32, 35, 37, 39], [1000, 1001, 857, 1]]);**

**Confirmar el final**

Compruebe si una cadena (primer argumento, str) termina con la cadena de destino dada (segundo argumento, target).

Este desafío se puede resolver con el .endsWith()método, que se introdujo en ES2015. Pero para el propósito de este desafío, nos gustaría que use uno de los métodos de subcadena de JavaScript en su lugar.

**function confirmEnding(str, target) {**

**let m = str.length;**

**let n = target.length;**

**for (let i = 0; i < n; i++) {**

**if (str[m - i - 1] != target[n - i - 1]){**

**return false**

**}**

**}**

**return true;**

**}**

**confirmEnding("Bastian", "n");**

**Repetir una cadena Repetir una cadena**

Repetir una cadena dada str(primer argumento) por numtiempos (segundo argumento). Devuelve una cadena vacía si numno es un número positivo. A los efectos de este desafío, no utilice el método incorporado .repeat().

**function repeatStringNumTimes(str, num) {**

**let newStr = "";**

**if (num > 0) {**

**for (let i = 0; i < num; i++){**

**newStr += str**

**}**

**}**

**return newStr;**

**}**

**repeatStringNumTimes("abc", 3);**

**Truncar una cadena**

Trunca una cadena (primer argumento) si es más larga que la longitud máxima de cadena dada (segundo argumento). Devuelve la cadena truncada con un ...final.

**function truncateString(str, num) {**

**if (str.length > num) {**

**return str.slice(0,num) + '...'**

**} else {**

**return str;**

**}**

**}**

**truncateString("A-tisket a-tasket A green and yellow basket", 8);**

**Buscadores Guardianes**

Cree una función que mire a través de una matriz arry devuelva el primer elemento que pase una 'prueba de verdad'. Esto significa que dado un elemento x, la 'prueba de verdad' se pasa si func(x)es true. Si ningún elemento pasa la prueba, devuelve undefined.

**function findElement(arr, func) {**

**return arr.find(func);**

**}**

**findElement([1, 2, 3, 4], num => num % 2 === 0);**

**abucheo quien**

Comprueba si un valor se clasifica como un primitivo booleano. Regresar trueo false.

Las primitivas booleanas son truey false.

**function booWho(bool) {**

**return typeof(bool) === 'boolean';**

**}**

**booWho(null);**

**Título Caso una Oración**

Devuelve la cadena proporcionada con la primera letra de cada palabra en mayúscula. Asegúrate de que el resto de la palabra esté en minúsculas.

A los efectos de este ejercicio, también debe escribir en mayúsculas las palabras de conexión como they of.

**function titleCase(str) {**

**const newTitle = str.split(" ");**

**const updatedTitle = [];**

**for (let st in newTitle) {**

**updatedTitle[st] = newTitle[st][0].toUpperCase() + newTitle[st].slice(1).toLowerCase();**

**}**

**return updatedTitle.join(" ");**

**}**

**titleCase("I'm a little tea pot");**

**Cortar y empalmar**

**Se le dan dos matrices y un índice.**

**Copie cada elemento de la primera matriz en la segunda matriz, en orden.**

**Comience a insertar elementos en el índice nde la segunda matriz.**

**Devuelve la matriz resultante. Las matrices de entrada deben permanecer iguales después de que se ejecute la función.**

**function frankenSplice(arr1, arr2, n) {**

**let newArr = arr2.slice(0);**

**newArr.splice(n, 0, ...arr1);**

**return newArr;**

**}**

**frankenSplice([1, 2, 3], [4, 5, 6], 1);**

**Gorila falso**

Elimina todos los valores falsos de una matriz. Devuelve una nueva matriz; no cambie la matriz original.

Los valores falsos en JavaScript son false, null, 0, "", undefinedy NaN.

Sugerencia: Intente convertir cada valor a un valor booleano.

**function bouncer(arr) {**

**const filteredArr = [];**

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

**if (arr[i]) filteredArr.push(arr[i]);**

**}**

**return filteredArr;**

**}**

**bouncer([7, "ate", "", false, 9]);**

**Dónde pertenezco**

Devuelve el índice más bajo en el que se debe insertar un valor (segundo argumento) en una matriz (primer argumento) una vez que se ha ordenado. El valor devuelto debe ser un número.

Por ejemplo, getIndexToIns([1,2,3,4], 1.5)debería regresar 1porque es mayor que 1(índice 0), pero menor que 2(índice 1).

Del mismo modo, getIndexToIns([20,3,5], 19)debería regresar 2porque una vez que se haya ordenado la matriz, se verá como [3,5,20]y 19es menor que 20(índice 2) y mayor que 5(índice 1).

**function getIndexToIns(arr, num) {**

**return arr.filter(val => num > val).length;**

**}**

**getIndexToIns([40, 60], 50);**

**Mutaciones**

Retorna truesi la cadena en el primer elemento de la matriz contiene todas las letras de la cadena en el segundo elemento de la matriz.

Por ejemplo, ["hello", "Hello"]debería regresar trueporque todas las letras en la segunda cadena están presentes en el primer caso, ignorando.

Los argumentos ["hello", "hey"]deberían regresar falseporque la cadena hellono contiene un y.

Por último, ["Alien", "line"]debería volver trueporque todas las letras de lineestán presentes en Alien.

**function mutation(arr) {**

**const test = arr[1].toLowerCase();**

**const target = arr[0].toLowerCase();**

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

**if (target.indexOf(test[i]) < 0) return false;**

**}**

**return true;**

**}**

**mutation(["hello", "hey"]);**

**mono fornido**

Escriba una función que divida una matriz (primer argumento) en grupos de la longitud de size(segundo argumento) y los devuelva como una matriz bidimensional.

**function chunkArrayInGroups(arr, size) {**

**// Break it up.**

**const newArr = [];**

**for (let i = 0; i < arr.length; i += size) {**

**newArr.push(arr.slice(i, i + size));**

**}**

**return newArr;**

**}**

**chunkArrayInGroups(["a", "b", "c", "d"], 2);**