**Prueba 1**

**Comprobador de palíndromo**

Devuelve truesi la cadena dada es un palíndromo. De lo contrario, regresa false.

Un palíndromo es una palabra u oración que se escribe de la misma manera hacia adelante y hacia atrás, ignorando la puntuación, el caso y el espacio.

Nota: deberá eliminar todos los caracteres no alfanuméricos (puntuación, espacios y símbolos) y convertir todo en el mismo caso (minúsculas o mayúsculas) para verificar si hay palíndromos.

Pasaremos cadenas con diferentes formatos, como racecar, RaceCary race CARentre otros.

También pasaremos cadenas con símbolos especiales, como 2A3\*3a2, 2A3 3a2y 2\_A3\*3#A2.

**function orbitalPeriod(arr) {**

**const GM = 398600.4418;**

**const earthRadius = 6367.4447;**

**const newArr = [];**

**//Looping through each key in arr object**

**for (let elem in arr) {**

**//Rounding off the orbital period value**

**const orbitalPer = Math.round(**

**2 \* Math.PI \* Math.sqrt(Math.pow(arr[elem].avgAlt + earthRadius, 3) / GM)**

**);**

**//Adding new object with orbitalPeriod property**

**newArr.push({name: arr[elem].name, orbitalPeriod: orbitalPer});**

**}**

**return newArr;**

**}**

**// test here**

**orbitalPeriod([{ name: "sputnik", avgAlt: 35873.5553 }]);**

**Conversor de numeros romanos**

Convierte el número proporcionado en un número romano.

numeros romanos numeros arabigos

METRO 1000

CM 900

D 500

CD 400

C 100

XC 90

L 50

SG 40

X 10

IX 9

V 5

IV 4

I 1

Todas las respuestas de los números romanos deben ser proporcionados en mayúsculas.

**function convertToRoman(num) {**

**const numerals = {**

**1: 'I',**

**4: 'IV',**

**5: 'V',**

**9: 'IX',**

**10: 'X',**

**40: 'XL',**

**50: 'L',**

**90: 'XC',**

**100: 'C',**

**400: 'CD',**

**500: 'D',**

**900: 'CM',**

**1000: 'M',**

**};**

**let romanized = '';**

**const decimalKeys = Object.keys(numerals).reverse();**

**decimalKeys.forEach(key => {**

**while (key <= num){**

**romanized += numerals[key];**

**num -= key;**

**}**

**});**

**return romanized;**

**}**

**console.log(convertToRoman(36));**

**Cifrado César**

Uno de los grabados más simples y conocidos es el grabado César , también conocido como grabado por desplazamiento . En un cifrado por desplazamiento los significados de las letras se desplazan por una determinada cantidad.

Un uso moderno común es el grabador ROT13 , donde los valores de las letras son desplazados por 13 lugares. Así que A ↔ N, B ↔ Oy así sucesivamente.

Escribe una función que recibe una cadena codificada en ROT13 como entrada y devuelve una cadena decodificada.

Todas las letras estarán en mayúsculas. No transformes ningún carácter no alfabético (espacios, puntuación, por ejemplo), pero si transmítelos.

**function rot13(str) {**

**var cracker = {**

**A:'N', B:'O', C:'P', D:'Q', E:'R', F:'S', G:'T', H:'U', I:'V', J:'W', K:'X', L:'Y', M:'Z', N:'A', O:'B', P:'C', Q:'D', R:'E', S:'F', T:'G', U:'H', V:'I', W:'J', X:'K', Y:'L', Z:'M', ' ':' ', '.':'.', '!': '!', '?': '?'**

**};**

**return str.split('').map(char => cracker[char]).join('');**

**}**

**rot13("SERR PBQR PNZC");**

**Validador de numeros telefonos**

Devuelve truesi la cadena pasada coincide con un número de teléfono válido en Estados Unidos.

El usuario puede completar el campo del formulario de la forma que elija, siempre que tenga el formato de un número estadounidense válido. Los siguientes ejemplos son de formatos válidos para números estadounidenses (consulte las pruebas a continuación para otras variantes):

**555-555-5555**

**(555)555-5555**

**(555) 555-5555**

**555 555 5555**

**5555555555**

**1 555 555 5555**

Para este desafío se te precede una cadena como 800-692-7753o 8oo-six427676;laskdjf. Tu trabajo es validar o rechazar el número de teléfono estadounidense basado en cualquier combinación de los formatos proporcionados arriba. El codigo de area es obligatorio. Si el código de país es adecuado, debe confirmar que el código de país es 1. Devuelve truesi la cadena es un número de teléfono estadounidense válido; de lo contrario devuelve false.

**function telephoneCheck(str) {**

**let rex1 = /^(1\s?)?\d{3}([-\s]?)\d{3}\2\d{4}$/,**

**rex2 = /^(1\s?)?\(\d{3}\)\s?\d{3}[-\s]?\d{4}$/;**

**if (rex1.test(str)) {**

**return true;**

**} else {**

**return rex2.test(str) ? true : false**

**}**

**}**

**telephoneCheck("555-555-5555");**

**Caja registradora**

Diseñe una función de cajón de caja registradora checkCashRegister()que acepte el precio de compra como primer argumento ( price), el pago como segundo argumento ( cash) y el efectivo en cajón ( cid) como tercer argumento.

cides una matriz 2D que enumera la moneda disponible.

La checkCashRegister()función siempre debe devolver un objeto con una statusclave y una changeclave.

Devuélvalo {status: "INSUFFICIENT\_FUNDS", change: []}si el efectivo en el cajón es menor que el cambio adeudado, o si no puede devolver el cambio exacto.

Devolución {status: "CLOSED", change: [...]}con efectivo en cajón como valor de la llave changesi es igual al cambio adeudado.

En caso contrario, devolver {status: "OPEN", change: [...]}, con el cambio adeudado en monedas y billetes, ordenados de mayor a menor, como valor de la changellave.

Unidad monetaria Cantidad

Centavo $0.01 (PENIQUE)

Níquel $0.05 (NÍQUEL)

Diez centavos $0.1 (DIME)

Cuarto $0.25 (TRIMESTRE)

Dólar $1 (UNO)

Cinco dólares $5 (CINCO)

Diez dólares $10 (DIEZ)

Veinte dólares $20 (VEINTE)

Cien dolares $100 (CIEN)

Vea a continuación un ejemplo de una matriz de efectivo en cajón:

[

["PENNY", 1.01],

["NICKEL", 2.05],

["DIME", 3.1],

["QUARTER", 4.25],

["ONE", 90],

["FIVE", 55],

["TEN", 20],

["TWENTY", 60],

["ONE HUNDRED", 100]

]

**function checkCashRegister (price, cash, cid) {**

**var change;**

**let ret = cash - price;**

**const total = cid.reduce((p, c) => accAdd(p, c[1]), 0);**

**if (ret == total) {**

**change = { status: "CLOSED", change: cid }**

**} else if (ret > total) {**

**change = { status: "INSUFFICIENT\_FUNDS", change: [] }**

**} else {**

**const M1 = {**

**PENNY: 0.01,**

**NICKEL: 0.05,**

**DIME: 0.1,**

**QUARTER: 0.25,**

**ONE: 1,**

**FIVE: 5,**

**TEN: 10,**

**TWENTY: 20,**

**"ONE HUNDRED": 100,**

**}**

**const M2 = cid.reduce((p, c) => { p[c[0]] = c[1]; return p; }, {})**

**const retChange = [];**

**["ONE HUNDRED", "TWENTY", "TEN", "FIVE", "ONE", "QUARTER", "DIME", "NICKEL", "PENNY"].forEach(unit => {**

**let ret1;**

**if (!+ret) return;**

**if (M2[unit] && (ret1 = Math.floor(accDiv(ret, M1[unit])))) {**

**const ret2 = accMul(ret1, M1[unit])**

**if (M2[unit] >= ret2) {**

**retChange.push([unit, ret2])**

**ret = accSub(ret, ret2)**

**} else {**

**retChange.push([unit, M2[unit]])**

**ret = accSub(ret, M2[unit])**

**}**

**}**

**})**

**if (+ret) {**

**change = { status: "INSUFFICIENT\_FUNDS", change: [] }**

**} else {**

**change = { status: "OPEN", change: retChange }**

**}**

**}**

**return change;**

**}**

**function accAdd (arg1, arg2) {**

**var r1, r2, m, c;**

**try {**

**r1 = arg1.toString().split(".")[1].length;**

**}**

**catch (e) {**

**r1 = 0;**

**}**

**try {**

**r2 = arg2.toString().split(".")[1].length;**

**}**

**catch (e) {**

**r2 = 0;**

**}**

**c = Math.abs(r1 - r2);**

**m = Math.pow(10, Math.max(r1, r2));**

**if (c > 0) {**

**var cm = Math.pow(10, c);**

**if (r1 > r2) {**

**arg1 = Number(arg1.toString().replace(".", ""));**

**arg2 = Number(arg2.toString().replace(".", "")) \* cm;**

**} else {**

**arg1 = Number(arg1.toString().replace(".", "")) \* cm;**

**arg2 = Number(arg2.toString().replace(".", ""));**

**}**

**} else {**

**arg1 = Number(arg1.toString().replace(".", ""));**

**arg2 = Number(arg2.toString().replace(".", ""));**

**}**

**return (arg1 + arg2) / m;**

**}**

**function accSub (arg1, arg2) {**

**var r1, r2, m, n;**

**try {**

**r1 = arg1.toString().split(".")[1].length;**

**}**

**catch (e) {**

**r1 = 0;**

**}**

**try {**

**r2 = arg2.toString().split(".")[1].length;**

**}**

**catch (e) {**

**r2 = 0;**

**}**

**m = Math.pow(10, Math.max(r1, r2)); //last modify by deeka //动态控制精度长度**

**n = (r1 >= r2) ? r1 : r2;**

**return ((arg1 \* m - arg2 \* m) / m).toFixed(n);**

**}**

**function accMul (arg1, arg2) {**

**var m = 0, s1 = arg1.toString(), s2 = arg2.toString();**

**try {**

**m += s1.split(".")[1].length;**

**}**

**catch (e) {**

**}**

**try {**

**m += s2.split(".")[1].length;**

**}**

**catch (e) {**

**}**

**return Number(s1.replace(".", "")) \* Number(s2.replace(".", "")) / Math.pow(10, m);**

**}**

**function accDiv (arg1, arg2) {**

**var t1 = 0, t2 = 0, r1, r2;**

**try {**

**t1 = arg1.toString().split(".")[1].length;**

**}**

**catch (e) {**

**}**

**try {**

**t2 = arg2.toString().split(".")[1].length;**

**}**

**catch (e) {**

**}**

**r1 = Number(arg1.toString().replace(".", ""));**

**r2 = Number(arg2.toString().replace(".", ""));**

**return (r1 / r2) \* Math.pow(10, t2 - t1);**

**}**

**checkCashRegister(3.26, 100, [["PENNY", 1.01], ["NICKEL", 2.05], ["DIME", 3.1], ["QUARTER", 4.25], ["ONE", 90], ["FIVE", 55], ["TEN", 20], ["TWENTY", 60], ["ONE HUNDRED", 100]])**