1. Suma de digitos

Dado un *integer*, escribe una función que regrese la suma todos sus dígitos. Es decir, para el *integer* 12345 las suma de sus dígitos es: 1+2+3+4+5 = 15

Definición de la función

+ digitsSum(inputInt: integer): integer

Casos de ejemplo

- Con inputInt = “999”, la salida debería ser digitsSum(inputInt) = 27;
- Con inputInt = “9184501”, la salida debería ser digitsSum(inputInt) = 28; ● Con inputInt = “12345”, la salida debería ser digitsSum(inputInt) = 15;

```
const inputInt = (numero) => {
    //Convertimos la entrada a string
    const cadena = numero.toString();
    //creamos un array
    let array=[];
    //Agregamos cada valor del numero a un array
    for (let i = 0; i < cadena.length; i++) {
        array.push(cadena[i]);
    }
    let contador=0;
    for (let j = 0; j < cadena.length; j++){
        contador+=parseInt(cadena[j]);
    }
    console.log(contador);
}
inputInt(999);
```

```
[nodemon] starting `node index.js`
27
[nodemon] clean exit - waiting for changes before restart
```

2. Palíndromos

Dado un *string*, escribe una función para verificar si es un palíndromo. Un palíndromo es un texto que se lee igual de izquierda a derecha que de derecha a izquierda. Las palabras: **salas, oso, reconocer y oro** son palíndromos.

Definición de la función

+ isPalindrome(inputStr: string): boolean

Casos de ejemplo

- Con inputStr = “aabaa”, la salida debería ser isPalindrome(inputStr) = true;
- Con inputStr = “abac”, la salida debería ser isPalindrome(inputStr) = false; ● Con inputStr = “salas”, la salida debería ser isPalindrome(inputStr) = true;

```
const esPalindromo = (cadena) => {
    //pasamos la cadena a minusculas
    const cadenaRevisar =cadena.toLowerCase();
    let bandera;
    //creamos una cadena invertida para evaluarla despues
    const cadenaInvertida = cadenaRevisar.split("").reverse().join("");
    if (cadenaInvertida === cadenaRevisar){
        bandera=true;
        return console.log(bandera);
    }else{
        bandera=false;
        return console.log(bandera);
    }

}

esPalindromo("aabaa");
/*
[nodemon] starting `node index.js`
true
[nodemon] clean exit - waiting for changes before restart
```

3. Producto de elementos adyacentes

Dado un arreglo de enteros, encuentra el par de elementos adyacentes tales que **su producto sea el más grande** y devuelve dicho producto.

Se consideran como elementos adyacentes aquellos que se encuentren a la izquierda o a la derecha, es decir, dado el arreglo [3, 6, -2, 5] los elementos adyacentes serían:

- 3 y 6 => 3 * 6 = **18**
- 6 y-2 => 6 * -2 = -12
- -2 y 5 => 5 * -2 = -10

La función debería devolver **18** dado que es el producto de adyacentes más grande.

Definición de la función

+ maxAdjacentProd(inputArray: array): integer

Casos de ejemplo

- Con inputArray= [3 , 6, -2, -5, 7, 3] la salida debería ser maxAdjacentProd(inputArray) = 21;

- Con inputArray=[5 , 1, 2, 3, 1, 4] la salida debería ser maxAdjacentProd(inputArray) = 6;

- Con inputArray=[-23 , 4, -3, 8, -12] la salida debería ser maxAdjacentProd(inputArray) = -12;

```
const inputArray = (array) => {
    //variable temporal para guardar el valor mas grande
    let temp=Number.NEGATIVE_INFINITY;
    //Iteramos en el array
    for (let i=0; i< array.length-1; i++){
        //Si el valor de la multiplicacion es mayor que el valor guardado
        if(array[i]*array[i+1] > temp){
            //guardamos la salida
            temp=array[i]*array[i+1];
        }
    }
    console.log(temp);
}
inputArray([-23 , 4, -3, 8, -12]);
```

```
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
-12
[nodemon] clean exit - waiting for changes before restart
```