**Fundamentos Intermedios**

**1.-Sigma -**Implementa una función sigma(num) que, dado un número, devuelve la suma de todos los enteros positivos (incluyendo el número dado). Ej: sigma(3) = 6 (1+2+3); sigma(5) = 15 (1+2+3+4+5).

function sigma(num){

var sumatoria =0;

for(var i=1; i<=num; i++){

    sumatoria=sumatoria +i,

}

return sumatoria;

}

var a =sigma(1);

console.log(a);

**2.-Factorial -**Escribe una función factorial(num) que, dado un número, devuelva el producto (multiplicación) de todos los enteros positivos (incluyendo el número dado). Por ejemplo: factorial(3) = 6 (1\*2\*3); factorial(5) = 120 (1\*2\*3\*4\*5).

function factorial(num){

var f =1;

for (var i=1; i<=num; i++){

    f=f\*i;

}

return f;

}

var a = factorial (5);

console.log(a);

**3.-Fibonacci -**Crea una función para generar números de Fibonacci. En esta famosa secuencia matemática, cada número es la suma de las dos anteriores, partiendo con los valores 0 y 1. Tu función debería aceptar un argumento, un índice en la secuencia (donde 0 corresponde al valor inicial, 4 corresponden al valor cuatro más tarde, etc). Ejemplos: fibonacci(0) = 0 (dado), fibonacci(1) = 1 (dado), fibonacci(2) = 1 (fib(0)+fib(1), o 0+1), fibonacci(3) = 2 (fib(1) + fib(2)3, o 1+1), fibonacci(4) = 3 (1+2), fibonacci(5) = 5 (2+3), fibonacci(6) = 8 (3+5), fibonacci(7) = 13 (5+8). Haz esto primero sin usar recursión. Si no sabes qué es una recursión, no te preocupes puesto que vamos a introducir este concepto en la Parte 2 de esta actividad.

function fibonacci(x){

    var anterior0 =0

    var anterior1 =1

    var fibo =0

        console.log(anterior0);

        console.log(anterior1);

    for(var i=0; i<x-1; i++){

        fibo=anterior0 + anterior1;

        console.log (fibo);

        anterior0=anterior1;

        anterior1= fibo;

    }

}

f=fibonacci(7);

**4.-Array: Penúltimo:** Devuelve el penúltimo elemento del array. Dado [42,true,4,”Liam”, 7] devuelve “Liam”. Si el array es muy pequeño, devuelve null.

var arreglo =[42,true,4,"Lian",7]

   console.log(arreglo[arreglo.length-2]);

**5.-Array: “N” último:** Devuelve el elemento “N” último. Dado ([5,2,3,6,4,9,7],3], devuelve 4. Si el array es muy pequeño, devuelve null.

function n\_ultimo(arreglo,n){

    return arreglo[arreglo.length-n];

     var arreglo =[5,2,3,6,4,9,7];

     var n=3;

}

**6.-Array: Segundo más grande:**Devuelve el segundo elemento más grande de un array. Dado [42,1,4,3.14,7], devuelve 7.  Si el array es muy pequeño, devuelve null.

function masgrande(x){

    var masgrande1 =0;

    var masgrande2=0;

    if (x.length<2){

        return null;

    }

    for (var i= 0; i<= x.length; i++){/\* solo para detectar el + grande\*/

        if(masgrande1<=x[i]){

            masgrande1=x[i];

        }

    }

    for (var i= 0; i<= x.length; i++){/\* detecta el 2 mas grande\*/

        if(masgrande1==x[i]){/\* mas grande =0\*/

        x[i]=0;

        }

        if(masgrande2<x[i]){ /\* detectar el nuevo mas grande\*/

            masgrande2=x[i];

        }

    }

        return masgrande2;

}

x = [42,1,4,3.14,7];

console.log(masgrande(x))

**7.-Doble Problema Par:**Crea una función que cambie un array dado duplicando cada uno de sus elementos en una posición par, y manteniendo el orden original. Convierte [4, "Ulysses", 42, false]   a    [4,4, "Ulysses", 42, 42, false].

function duplica\_par(x){//x=[4,2]

    var nuevo\_arreglo = [4,4];

    for (var index = 0; index < x.length; index++) {//index= index+1

        nuevo\_arreglo.push(x[index]);

        if(index%2==0){//Modulo o resto de la division

            nuevo\_arreglo.push(x[index]);

        }

    }

    return nuevo\_arreglo;

}

var z = duplica\_par([4, "Ulysses", 42, false]);

console.log(z);