/\*Sigma - Implement function sigma(num) that, given a number, returns the sum of all positive integers up to the given number (inclusive).

Ex: sigma(3) = 6 (or 1+2+3); sigma(5) = 15 (or 1+2+3+4+5).\*/

function sigma(num){

var sum = 0;

for(let i = 1; i <= num ; i++){

sum += i;

}

return sum;

}

console.log( sigma(5));

/\*Factorial - Write a function factorial(num) that, given a number, returns the product (multiplication) of all positive integers from 1

 up to the given number (inclusive).  For example, factorial(3) = 6 (or 1\*2\*3); factorial(5) = 120 (or 1\*2\*3\*4\*5).\*/

 function factorial(num){

    var mult = 1;

    for(let i = 1; i <= num ; i++){

    mult \*= i;

    }

    return mult;

    }

    console.log( factorial(5));

/\*Fibonacci - Create a function to generate Fibonacci numbers.  In this famous mathematical sequence, each number

is the sum of the previous two, starting with values 0 and 1.  Your function should accept one argument, a

n index into the sequence (where 0 corresponds to the initial value, 4 corresponds to the value four later, etc).

  Examples: fibonacci(0) = 0 (given), fibonacci(1) = 1 (given), fibonacci(2) = 1 (fib(0)+fib(1), or 0+1), fibonacci(3) = 2

  (fib(1) + fib(2)3, or 1+1), fibonacci(4) = 3 (1+2), fibonacci(5) = 5 (2+3), fibonacci(6) = 8 (3+5), fibonacci(7) = 13 (5+8).

    Do this without using recursion first.  If you

 don't know what a recursion is yet, don't worry as we'll be introducing this concept in Part 2 of this assignment.\*/

 function Fibonacci(num) {

var arr = [0,1]

for(var i=2 ; i<=num ; i++){

arr.push(arr[i-1]+arr[i-2]);

}

console.log(arr)

console.log(arr[arr.length-1])

}

//Fibonacci(7);

/\*Array: Second-to-Last: Return the second-to-last element of an array. Given [42, true, 4, "Liam",

 7], return "Liam".  If array is too short, return null.\*/

 function Second\_to\_Last(arr){

    if(arr.length<=1){

    //console.log(5)

    return null

    }

    return arr[arr.length-2];

    }

    //console.log(Second\_to\_Last([1,2,3]))

    /\*Array: Nth-to-Last: Return the element that is N-from-array's-end.  Given ([5,2,3,6,4,9,7],3), return 4.

      If the array is too short, return null.\*/

      function Nth\_to\_Last(arr,N){

        if(arr.length<N){

        return null;

        }

        else{

        return arr[arr.length-N];

        }

        }

        //console.log(Nth\_to\_Last([1,2,3,4,5],4))

        /\*Array: Second-Largest:

        Return the second-largest element of an array. Given [42,1,4,3.14,7], return 7.  If the array is too short,

        return null.\*/

        function Second\_Largest(arr){

            var max = 0;

            var max2= 0;

            var z;

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

                if(max<arr[i]){

                    max=arr[i];

                    z=i

                }

                }

                for(var j=0 ; j<=arr.length ; j++){

                    if(max2<arr[j]&&arr[j]!==arr[z]){

                        max2=arr[j];

            }

            }

            return max2;

           }

            var arr=[1,8,6,7]

        console.log(Second\_Largest(arr));

        /\*Double Trouble: Create a function that changes a given array to list each existing element

        twice, retaining original order.  Convert [4, "Ulysses", 42, false] to [4,4, "Ulysses", "Ulysses", 42, 42, false, false].\*/

         function twice(arr){

            var arr2=[];

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

             arr2.push(arr[i]);

             arr2.push(arr[i]);

            //console.log(arr2);

            }

            //console.log(arr2);

            return arr2;

            }

            var arr = [1,2,3,4];

            console.log(twice(arr));