// print 1-n with normal recursion

let n = 4;

var f = function(i,n){

    if(i>n) return;

    console.log(i);

    f(i+1,n);

}

f(1,n);

// print the number 1-N using backtracking

let n = 4;

var f = function (i, n) {

  if (i < 1) {

    return;

  }

  f(i-1,n); // first call the function then print the number

  console.log(i);

};

f(n, n);

// print N to 1 using backtracking

var f = function(i,n){

    if(i>n) return;

    f(i+1,n);

    console.log(i);

}

let n = 10;

f(1,n);

// summation of numbers from 1-N using parameterized method

// in parameterrized method, we pass the needed parameter as a param in the next function call

// in below example we are doing summation from N to 1

var summation = function(i,sum){

    if(i==0){

        console.log(sum);

        return;

    }

    summation(i-1,sum+i);

}

let num = 5;

summation(num,0);

// summation using function returning a value

var summation = function(n){

    if(n==0){

        return 0;

    }

    return n + summation(n-1);

}

console.log(summation(5));

// factorial of a number using recursion

var fact = function(n){

    if(n==1){

        return 1;

    }

    return n\*fact(n-1);

}

console.log(fact(5));

// reverse the array with recursion and two pointer approach

var reverse = function (arr, left, right) {

  if (left >= right) {

    return;

  }

  let temp = arr[left];

  arr[left] = arr[right];

  arr[right] = temp;

  left++;

  right--;

  reverse(arr, left, right);

};

let arr = [1, 2, 3, 4, 5, 6];

let res = reverse(arr, 0, arr.length - 1);

console.log(arr);

// reverse the array using single pointer and recursion

function reverseSinglePointer(arr, i, n) {

    console.log("arr", arr, "i", i, "n", n);

    if (i >= n / 2) return;

    let temp = arr[i];

    arr[i] = arr[n - i - 1];

    arr[n - i - 1] = temp;

    i++;

    reverseSinglePointer(arr, i, n);

  }

  let arrSample = [1, 2, 3, 4, 5, 6];

  reverseSinglePointer(arrSample, 0, arrSample.length);

  console.log(arrSample);

// check palindrome wit single pointer recursion

function pal(i,str,n){

    if(i>=n/2) return true;

    if(str[i]!=str[n-i-1]) return false;

    return pal(i+1,str,n);

}

let str = 'aaaa';

let tempArr = str.split('');

console.log(pal(0,tempArr,tempArr.length));

// find fibonacci with multiple recursion

// 0 1 1 2 3 5 8

function f(n){

    if(n<=1) {

        // console.log(n)

        return n;

    }

    let flast = f(n-1);

    let secLast = f(n-2);

    return flast + secLast;

}

console.log(f(6));

// time complexity : 2(n);

