# Solutions-Exercise: Arrays and Matrices

## 1.Print an Array with a given Delimiter

function printWithDelimiter(arr){

let delimiter = arr.pop();

console.log(arr.join(delimiter));

}

printWithDelimiter(['One', 'Two', 'Three', 'Four', 'Five', '-'])

printWithDelimiter(['How about no?', 'I', 'will', 'not', 'do', 'it!', '\_'])

|  |
| --- |
| function solve(array){ |
|  | let delimiter = array[array.length - 1]; |
|  | let result = ""; |
|  | array.pop(); |
|  | for(let i = 0; i < array.length;i++){ |
|  | if(i === 0){ |
|  | result += array[i] |
|  | }else { |
|  | result += delimiter + array[i]; |
|  | } |
|  | } |
|  | console.log(result); |
|  | } |

## 2.Print every N-th Element from an Array

function printEveryNthElement (arr){

let step = Number(arr.pop());

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

console.log(arr[i]);

}

}

printEveryNthElement(['5', '20', '31', '4', '20', '2'])

printEveryNthElement(['dsa', 'asd', 'test', 'tset', '2'])

printEveryNthElement(['1', '2', '3', '4', '5', '6'])

|  |
| --- |
| function solve(array){ |
|  | let step = Number(array[array.length-1]); |
|  | array.pop(); |
|  | for(let i = 0;i<array.length;i += step){ |
|  | console.log(array[i]); |
|  | } |
|  | } |

## 3.\*Add and Remove Elements from an Array

function addAndRemoveElements(arr){

let resultArr = arr.reduce((acc, el, idx) =>{

if (el === 'add'){

acc.push(idx + 1);

} else if (el === 'remove'){

acc.pop();

}

return acc;

}, []);

if (resultArr.length > 0){

console.log(resultArr.join('\n'));

} else {

console.log('Empty');

}

}

addAndRemoveElements(['add', 'add', 'add', 'add'])

addAndRemoveElements(['add', 'add', 'remove', 'add', 'add'])

addAndRemoveElements(['remove', 'remove', 'remove'])

|  |
| --- |
| function solve(array){ |
|  | let result = []; |
|  | let n = 0; |
|  | for(let i = 0; i < array.length;i++){ |
|  | let command = array[i]; |
|  | n++; |
|  |  |
|  | if(command === "add"){ |
|  | result.push(n); |
|  | }else{ |
|  | result.pop(); |
|  | } |
|  | } |
|  | if(result.length === 0){ |
|  | result.push('Empty'); |
|  | } |
|  | for(let j = 0; j < result.length;j++){ |
|  | console.log(result[j]); |
|  | } |
|  | } |

## 4.Rotate Array

function rotateArray(arr){

let rotationCount = Number(arr.pop());

for (let i = 0; i < rotationCount % arr.length; i++) {

arr.unshift(arr.pop());

}

console.log(arr.join(' '));

}

rotateArray(['1', '2', '3', '4', '2'])

rotateArray(['Banana', 'Orange', 'Coconut', 'Apple', '15'])

|  |
| --- |
| function solve(array){ |
|  | let rotations = array[array.length-1]; |
|  | array.pop(); |
|  | for(let i = 1; i <= (+rotations % array.length);i++){ |
|  | let last = array[array.length-1]; |
|  | array.pop(); |
|  | array.unshift(last); |
|  | } |
|  | console.log(array.join(' ')); |
|  | } |

## 5.Extract a Non-decreasing Subsequence from an Array

function extractNondecreasingSubsecuence(arr){

let last = -Infinity;

let nondecreasingArr = arr.reduce((acc, el) => {

if (el >= last) {

acc.push(el);

last = el;

}

return acc;

}, []);

console.log(nondecreasingArr.join('\n'));

}

extractNondecreasingSubsecuence([1, 3, 8, 4, 10, 12, 3, 2, 24])

extractNondecreasingSubsecuence([1, 2, 3, 4])

extractNondecreasingSubsecuence([20, 3, 2, 15, 6, 1])

|  |
| --- |
| function solve(array) { |
|  | array = array.map(Number); |
|  | let result=[]; |
|  | let biggest = array[0]; |
|  | for (let i = 0; i < array.length; i++) { |
|  | if(array[i]>=biggest){ |
|  | result.push(array[i]); |
|  | biggest=array[i]; |
|  | } |
|  | } |
|  | console.log(result.join('\n')); |
|  | } |

## 6.Sort an Array by 2 Criteria

function sortArray(arr){

arr.sort((a, b) => {

return a.length - b.length || a.localeCompare(b)

});

console.log(arr.join('\n'));

}

sortArray(['alpha', 'beta', 'gamma'])

sortArray(['Isacc', 'Theodor', 'Jack', 'Harrison', 'George'])

sortArray(['test', 'Deny', 'omen', 'Default'])

sortArray(['Atest', 'Adeny', 'aomen', 'Default'])

|  |
| --- |
| function solve(array){ |
|  | array.sort(function(a, b){ |
|  | if(a.length === b.length){ |
|  | var nameA = a; |
|  | var nameB = b; |
|  | if (nameA < nameB) { |
|  | return -1; |
|  | } |
|  | if (nameA > nameB) { |
|  | return 1; |
|  | } |
|  | } |
|  | return a.length - b.length; |
|  | }); |
|  | console.log(array.join("\n")); |
|  | } |

## 7.Magic Matrices

function checkIfMatrixIsMagic(matrix){

let firstRowSum = 0;

for (let row = 0; row < matrix.length; row++) {

let rowSum = 0;

for (let col = 0; col < matrix[row].length; col++) {

rowSum += matrix[row][col];

}

if (row === 0){

firstRowSum = rowSum;

} else if (firstRowSum !== rowSum){

return false;

}

}

for (let col = 0; col < matrix[0].length; col++) {

let colSum = 0;

for (let row = 0; row < matrix.length; row++) {

colSum += matrix[row][col];

}

if (firstRowSum !== colSum){

return false;

}

}

return true;

}

console.log(checkIfMatrixIsMagic([[4, 5, 6], [6, 5, 4], [5, 5, 5]]))

console.log(checkIfMatrixIsMagic([[11, 32, 45], [21, 0, 1], [21, 1, 1]]))

console.log(checkIfMatrixIsMagic([[1, 0, 0], [0, 0, 1], [0, 1, 0]]))

|  |
| --- |
| function solve(matrix) { |
|  | let sum = 0; |
|  | matrix[0].forEach(x => sum += x); |
|  | for (let row = 1; row < matrix.length; row++) { |
|  | let rowSum = 0; |
|  | matrix[row].forEach(x => rowSum += x); |
|  | if (rowSum !== sum) return false; |
|  | } |
|  | for (let col = 0; col < matrix[0].length; col++) { |
|  | let colSum = 0; |
|  | for (let row = 0; row < matrix.length; row++) |
|  | colSum += matrix[row][col]; |
|  | if (colSum !== sum) return false; |
|  | } |
|  | return true; |
|  | } |

## 8.\*Spiral Matrix

function doSpiralMatrix(rows, cols){

let totalCellsCount = rows \* cols;

let spiralMatrix = [];

for (let i = 0; i < rows; i++) {

let rowArray = [];

for (let j = 0; j < cols; j++) {

rowArray.push(0);

}

spiralMatrix.push(rowArray);

}

let row = 0;

let col = 0;

let step = 0;

for (let i = 0; i < totalCellsCount;) {

while(col + step < cols){

i++;

spiralMatrix[row][col] = i;

col++;

}

col--;

row++;

while(row + step < rows){

i++;

spiralMatrix[row][col] = i;

row++;

}

row--;

col--;

while(col >= step){

i++;

spiralMatrix[row][col] = i;

col--;

}

col++;

row--;

step++;

while(row >= step){

i++;

spiralMatrix[row][col] = i;

row--;

}

row++;

col++;

}

for (let rowArray of spiralMatrix){

console.log(rowArray.join(' '));

}

}

doSpiralMatrix(5, 5)

doSpiralMatrix(3, 3)

doSpiralMatrix(10, 10)

|  |
| --- |
| function solve(n) { |
|  | var total = n\*n; |
|  | var result= []; |
|  |  |
|  | for(var i=0;i<n;i++) { |
|  | var rs = []; |
|  | for(var j=0;j<n;j++) { |
|  | rs.push(0); |
|  | } |
|  | result.push(rs); |
|  | } |
|  |  |
|  | var x=0; |
|  | var y=0; |
|  | var step = 0; |
|  | for(let i=0;i<total;){ |
|  | while(y+step<n){ |
|  | i++; |
|  | result[x][y]=i; |
|  | y++; |
|  |  |
|  | } |
|  | y--; |
|  | x++; |
|  |  |
|  | while(x+step<n){ |
|  | i++; |
|  | result[x][y]=i; |
|  | x++; |
|  | } |
|  | x--; |
|  | y--; |
|  |  |
|  | while(y>=step){ |
|  | i++; |
|  | result[x][y]=i; |
|  | y--; |
|  | } |
|  | y++; |
|  | x--; |
|  | step++; |
|  |  |
|  | while(x>=step){ |
|  | i++; |
|  | result[x][y]=i; |
|  | x--; |
|  | } |
|  | x++; |
|  | y++; |
|  | } |
|  | for (let row of result){ |
|  |  |
|  | console.log(row.join(' ')); |
|  |  |
|  | } |
|  | } |

## 9.\*\*Diagonal Attack

function checkDiagonalsAndPrintMatrix(stringArray) {

let matrix = stringArray.map(row => row.split(' ').map(Number));

let firstDiagonalSum = 0;

let secondDiagonalSum = 0;

for (let row = 0; row < matrix.length; row++) {

for (let col = 0; col < matrix[row].length; col++) {

if (row === col) {

firstDiagonalSum += matrix[row][col];

}

if (col === matrix[row].length - 1 - row){

secondDiagonalSum += matrix[row][col];

}

}

}

// for (let i = 0; i < matrix.length; i++) {

// firstDiagonalSum += matrix[i][i];

// secondDiagonalSum += matrix[i][matrix[i].length - 1 - i];

// }

if (firstDiagonalSum === secondDiagonalSum) {

for (let row = 0; row < matrix.length; row++) {

for (let col = 0; col < matrix[row].length; col++) {

if (row !== col && row !== matrix[row].length - 1 - col) {

matrix[row][col] = firstDiagonalSum;

}

}

}

}

printMatrix(matrix);

function printMatrix(matrix) {

for (let i = 0; i < matrix.length; i++) {

console.log(matrix[i].join(' '));

}

}

}

checkDiagonalsAndPrintMatrix(['5 3 12 3 1', '11 4 23 2 5', '101 12 3 21 10', '1 4 5 2 2', '5 22 33 11 1'])

checkDiagonalsAndPrintMatrix(['1 1 1', '1 1 1', '1 1 0'])

function checkDiagonalsAndPrintMatrix(stringArray) {

let matrix = stringArray.map(row => row.split(' ').map(Number));

let firstDiagonalSum = 0;

let secondDiagonalSum = 0;

// for (let row = 0; row < matrix.length; row++) {

// for (let col = 0; col < matrix[row].length; col++) {

// if (row === col) {

// firstDiagonalSum += matrix[row][col];

// }

// if (col === matrix[row].length - 1 - row){

// secondDiagonalSum += matrix[row][col];

// }

// }

// }

for (let i = 0; i < matrix.length; i++) {

firstDiagonalSum += matrix[i][i];

secondDiagonalSum += matrix[i][matrix[i].length - 1 - i];

}

if (firstDiagonalSum === secondDiagonalSum) {

for (let row = 0; row < matrix.length; row++) {

for (let col = 0; col < matrix[row].length; col++) {

if (row !== col && row !== matrix[row].length - 1 - col) {

matrix[row][col] = firstDiagonalSum;

}

}

}

}

printMatrix(matrix);

function printMatrix(matrix) {

for (let i = 0; i < matrix.length; i++) {

console.log(matrix[i].join(' '));

}

}

}

checkDiagonalsAndPrintMatrix(['5 3 12 3 1', '11 4 23 2 5', '101 12 3 21 10', '1 4 5 2 2', '5 22 33 11 1'])

checkDiagonalsAndPrintMatrix(['1 1 1', '1 1 1', '1 1 0'])

|  |
| --- |
| function solve(matrixRows) { |
|  | let matrix = matrixRows.map( |
|  | row => row.split(' ').map(Number)); |
|  |  |
|  | let sumFirstDiagonal = 0; |
|  | for (let i = 0; i < matrix.length; i++) { |
|  | sumFirstDiagonal += matrix[i][i]; |
|  | } |
|  | let sumSecondDiagonal = 0; |
|  | for (let j = 0; j < matrix.length; j++) { |
|  | sumSecondDiagonal += matrix[j][matrix.length-1-j]; |
|  | } |
|  |  |
|  | if (sumFirstDiagonal === sumSecondDiagonal){ |
|  | for (let q = 0; q < matrix.length; q++) { |
|  | for (let z = 0; z < matrix.length; z++) { |
|  | if( q !== z && q !== matrix.length-1-z) { |
|  | matrix[q][z] = sumFirstDiagonal; |
|  | } |
|  | } |
|  | } |
|  | printMatrix(matrix); |
|  | } |
|  | else |
|  | { |
|  | printMatrix(matrix); |
|  | } |
|  |  |
|  | function printMatrix(matrix) { |
|  | for (let i = 0; i < matrix.length; i++) { |
|  | console.log(matrix[i].join(' ')) |
|  | } |
|  | } |
|  | } |

## 10.\*Orbit

function showOrbit(arr){

let cols = arr[0];

let rows = arr[1];

let starRow = arr[2];

let starCol = arr[3];

let screenMatrix = [];

for (let row = 0; row < rows; row++) {

screenMatrix[row] = [];

for (let col = 0; col < cols; col++) {

screenMatrix[row][col] = Math.max(Math.abs(row - starRow), Math.abs(col - starCol)) + 1;

}

}

for(let row of screenMatrix){

console.log(row.join(' '));

}

}

showOrbit([4, 4, 0, 0])

showOrbit([5, 5, 2, 2])

showOrbit([3, 3, 2, 2])

|  |
| --- |
| function solve(array) { |
|  | let matrix = []; |
|  | let width = +array[0]; |
|  | let height = +array[1]; |
|  | let x = +array[2]; |
|  | let y = +array[3]; |
|  | for(let i=0; i<width; i++) { |
|  | matrix.push([]); |
|  | } |
|  | for(let row = 0; row< width; row++) { |
|  | for(let col=0; col<height; col++) { |
|  | matrix[row][col] = Math.max(Math.abs(row - x), Math.abs(col - y)) + 1; |
|  | } |
|  | } |
|  | console.log(matrix.map(row => row.join(" ")).join("\n")); |
|  | } |