# Lab: Arrays

Problems for in-class lab for the ["JavaScript Advanced" course @ SoftUni](https://softuni.bg/courses/js-advanced). Submit your solutions in the SoftUni judge system at <https://judge.softuni.bg/Contests/2752/Lab-Arrays>.

## Sum First Last

Write a function that calculates and prints the sum of the first and the last elements in an array.

The **input** comes **as array of string elements** holding numbers.

The **output** is the return value of your function.

### Example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| ['20', '30', '40'] | 60 | ['5', '10'] | 15 |

function solve(input){  
  
 let arr=[];  
  
  
 for(let index in input){  
 if(index%2===0){  
 arr.push(input[index]);  
 }  
 }  
  
 ***console***.log(arr.join(' '));  
  
  
}solve(['20', '30', '40']);

function solve(input){  
  
 let fist=***Number***(input.shift());  
 let sec=***Number***(input.pop());  
  
 let result=fist+sec;  
 ***console***.log(result);  
  
  
}solve(['20', '30', '40']);

function solve(input){  
  
 let result=0;  
  
 for(let index in input){  
  
 if(index%2===0){  
 result+=***Number***(input[index]);  
 }  
  
 }  
  
 ***console***.log(result);  
  
}solve(['20', '30', '40']);

## 2Even Position Element

Write a function that finds the elements at even positions in an array.

The **input** comes as **array of string** elements.

The **output** is the return value of your function. Collect all elements in a string, separated by space.

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| ['20', '30', '40'] | 20 40 | ['5', '10'] | 5 |

let even=function (input) {  
  
 let evenIndex=' ';  
 let oddIndex=' ';  
  
 for(let i=0;i<input.length;i++){  
  
 if(i%2===0){  
 evenIndex+=input[i];  
 evenIndex+=` `;  
 }else if(i%2!==0){  
 oddIndex+=input[i];  
 oddIndex+=` `;  
 }  
  
 }  
  
 ***console***.log(` even ${evenIndex}`);  
 ***console***.log(` odd ${oddIndex}`);  
  
 return input;  
  
}  
***console***.log(even(['20', '30', '40']));  
***console***.log(even([32,34,324,32,655]));

## 3Negative / Positive Numbers

Write a JS function that processes the elements in an array one by one and produces a new array. **Prepend** each **negative** element at the front of the result and **append** each **positive** (or 0) element at the end of the result.

The **input** comes as **array of number elements**.

The **output** is printed on the console, each element on a new line.

### Example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| [7, -2, 8, 9] | -2  7  8  9 | [3, -2, 0, -1] | -1  -2  3  0 |

function solve(array){  
  
 let res=array.sort(sorted);  
  
 ***console***.log(res.join(' '));  
  
 for(let nums of res){  
 ***console***.log(nums);  
 }  
  
  
 function sorted(a,b){  
 return a-b;  
 }  
  
  
}solve([7, -2, 8, 9]);

function solve(input){  
  
 let res=input.sort((a,b)=> {return b-a;} );  
  
  
 ***console***.log(res);  
  
  
  
}solve([3, -2, 0, -1]);

## 3Last K Numbers Sequence

You are given two integers **n** and **k**. Write a JS function that generates and prints the following sequence:

* The first element is 1
* Every following element equals the **sum** of the previous **k** elements
* The length of the sequence is **n** elements

The **input** comes as **two number arguments**. The first element represents the number **n**, and the second – the number **k**.

The **output** is printed on the console on a single line, separated by space.

### Example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 6, 3 | 1 1 2 4 7 13 | 8, 2 | 1 1 2 3 5 8 13 21 |

#### Explanation

The 2nd element (1) is the sum of the 3 elements before it, but there is only 1, so we take that. The third element is the sum of the first 2 (1 and 1) and the 4th – the sum of 1, 1 and 2. The 5th element is the sum of the 2nd, 3rd and 4th (1, 2 and 4) and so on.

function solve(n,k){  
  
let result=[1];  
  
 for(let i=1;i<n;i++){  
  
 let startIndex=***Math***.max(0,i-k);  
 let currentElement=result.slice(startIndex,startIndex+k).reduce((a,b)=>a+b,0);  
 result.push(currentElement);  
  
 }  
  
 ***console***.log(result.join(' '));  
  
}solve(6, 3);

## Process Odd Numbers

You are given an array of numbers. Write a JS function that prints the elements at **odd** **positions** from the array, **doubled** and in **reverse** order.

The **input** comes as **array of number elements**.

The **output** is printed on the console on a single line, separated by space.

### Example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| [10, 15, 20, 25] | 50 30 | [3, 0, 10, 4, 7, 3] | 6 8 0 |
|  |  |  |  |  |

function solve(input){  
  
 let even=0;  
 let odd=0;  
  
 for(let index in input){  
  
 if(index%2!==0){  
 odd+=input[index];  
 }else if(index%2===0){  
 even+=input[index];  
 }  
  
 }  
  
 ***console***.log(even);  
 ***console***.log(odd);  
  
}solve([10, 15, 20, 25]);

function solve(input){  
  
 let result=input  
 .filter((x,i)=> i%2!==0 )  
 .map(x=> x\*2 )  
 .reverse()  
 .join(' ');  
  
 ***console***.log(result)  
  
}solve([10, 15, 20, 25]);

## Smallest Two Numbers

Write a function that prints the two smallest elements from an array of numbers.

The **input** comes as **array of number elements**.

The **output** is printed on the console on a single line, separated by space.

### Example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| [30, 15, 50, 5] | 5 15 | [3, 0, 10, 4, 7, 3] | 0 3 |

function solve(input){  
  
  
 let array=input.sort((a,b)=>{  
  
 return a-b;  
  
 });  
  
 ***console***.log(array);  
  
 let biggest=array.slice(2);  
 ***console***.log(`The last biggest elements of array:`);  
 ***console***.log(biggest.join(' '));  
  
  
 let smallest=array.splice(0,2);  
 ***console***.log(`The first two smallest numbers of array:`);  
 ***console***.log(smallest.join(' '));  
  
}solve([30, 15, 50, 5]);

## Biggest Element

Write a function that finds the biggest element inside a matrix.

The **input** comes as **array of arrays**, containing number elements (2D matrix of numbers).

The **output** is the return value of your function. Find the biggest element and return it.

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| [[20, 50, 10],  [8, 33, 145]] | 145 | [[3, 5, 7, 12],  [-1, 4, 33, 2],  [8, 3, 0, 4]] | 33 |

function solve(input){  
  
 let arr=[];  
  
 for(let row of input){  
  
 for(let number of row){  
  
 arr.push(number);  
  
 }  
 }  
  
 ***console***.log(arr);  
  
 let biggest=arr.sort((a,b)=>{  
  
 return b-a;  
  
 });  
  
 ***console***.log(biggest);  
  
 let result=biggest.splice(0,1);  
 ***console***.log(`The biggest number is :`,result.join(' '));  
  
}solve([[20, 50, 10],  
 [8, 33, 145]]);

function solve(input){  
  
 let array=[];  
 for(let row of input){  
  
 for(let numbers of row){  
  
 array.push(numbers);  
  
 }  
  
 }  
  
 ***console***.log(array);  
  
 let sorted=array.sort(biggest);  
  
 ***console***.log(sorted);  
  
 let result=sorted.shift();  
 ***console***.log(`The biggest number is ${result} `)  
  
  
 function biggest(a,b) {  
  
 return b-a;  
  
 }  
  
  
}solve([[3, 5, 7, 12],  
 [-1, 4, 33, 2],  
 [8, 3, 0, 4]]);

## Diagonal Sums

A square matrix of numbers comes as an array of **strings**, each string holding numbers (space separated). Write a function that finds the sum at the main and at the secondary diagonals.

The **input** comes as **array of arrays**, containing number elements (2D matrix of numbers).

The **output** is printed on the console, on a single line separated by space. First print the sum at the main diagonal, then the sum at the secondary diagonal.

### Example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| [[20, 40],  [10, 60]] | 80 50 | [[3, 5, 17],  [-1, 7, 14],  [1, -8, 89]] | 1. 5 |

let result=function (input) {  
  
 let firstDiagonal=input[0][0]+input[1][1];  
  
  
  
  
 let secondDiagonal=input[0][1]+input[1][0];  
  
 ***console***.log(secondDiagonal);   
   
 return firstDiagonal;  
   
}  
  
***console***.log(result( [[20, 40],  
 [10, 60]]));

function solve(input){  
  
 let firstDiagonal=0;  
 let secondDiagonal=0;  
 let firstIndex=0;  
 let secondIndex=input[0].length-1;  
  
  
 input.forEach(row=>{  
 ***console***.log(row.join(' '));  
  
 })  
  
 input.forEach(array=>{  
  
 firstDiagonal+=array[firstIndex++];  
 secondDiagonal+=array[secondIndex--];  
  
 });  
  
  
 ***console***.log(` first is ${firstDiagonal}`)  
  
 ***console***.log(`second is ${secondDiagonal}`);  
  
}solve([[3, 5, 17],  
 [-1, 7, 14],  
 [1, -8, 89]]);

(function (array) {  
  
 let firstD=array[0][0]+array[1][1]+array[2][2];  
 ***console***.log(firstD);  
  
 let secondD=array[0][2]+array[1][1]+array[2][0];  
 ***console***.log(secondD);  
  
  
  
})([[3, 5, 17],  
 [-1, 7, 14],  
 [1, -8, 89]]);

## 7 Equal Neighbors

Write a function that finds the number of **equal** **neighbor** pairs inside a **matrix** of variable size and type (numbers or strings).

The **input** comes as **array of arrays**, containing string elements (2D matrix of strings).

The **output** is return value of your function. Save the number of equal pairs you find and return it.

### Example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| [['2', '3', '4', '7', '0'],  ['4', '0', '5', '3', '4'],  ['2', '3', '5', '4', '2'],  ['9', '8', '7', '5', '4']] | 1 | [['test', 'yes', 'yo', 'ho'],  ['well', 'done', 'yo', '6'],  ['not', 'done', 'yet', '5']] | 2 |

function equalNeighborsCount(matrix) {  
 let neighbors = 0;  
 for (let row = 0; row < matrix.length; row++) {  
 for (let col = 0; col < matrix[row].length; col++) {  
 if (row < matrix.length-1) {  
 if (matrix[row][col] == matrix[row + 1][col]) {  
 neighbors++;  
 }  
 }  
 if(col<matrix[row].length) {  
 if (matrix[row][col] == matrix[row][col + 1]) {  
 neighbors++;  
 }  
 }  
 }  
 }  
 ***console***.log(neighbors);  
}  
  
equalNeighborsCount([['2', '3', '4', '7', '0'],  
 ['4', '0', '5', '3', '4'],  
 ['2', '3', '5', '4', '2'],  
 ['9', '8', '7', '5', '4']]  
);