More Exercises: Arrays and Matrices

1. Sort an Array by 2 Criteria

Write a function that orders a **given array of strings** by **length** in **ascending order** as **primary criteria**, and by **alphabetical value** in **ascending order** as **secondary criteria**. The comparison should be **case-insensitive**.

The **input** comes as an **array of strings**.

The **output** is the ordered array of strings.

Examples

Input	Output
['alpha', 'beta', 'gamma']	beta alpha gamma

Input	Output
['Isacc', 'Theodor', 'Jack', 'Harrison', 'George']	Jack Isacc George Theodor Harrison

Input	Output
['test', 'Deny', 'omen', 'Default']	Deny omen test Default

Hints

- An array can be sorted by passing a comparing function to the **`Array.sort()`** function.
- Creating a comparing function by 2 criteria can be achieved by first comparing by the **main criteria**, if the 2 items are different (the result of the compare is not 0) - return the result as the result of the comparing function. If the two items are the same by the **main criteria** (the result of the compare is 0), we need to compare by the **second criteria** and the result of that comparison is the result of the comparing function.
- You can check more about **`Array.sort()`** here - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort

What to submit?

Function Signature: function main(strings)

Multidimensional Arrays (Matrices)

We will mainly work with 2-dimensional arrays. The concept is as simple as working with a simple 1-dimensional array. It is just an array of arrays.

2. Magic Matrices

Write a function that checks if a given matrix of numbers is **magical**. A matrix is magical if the **sums of the cells** of **every row** and **every column** are **equal**.

The **input** comes as an **array of arrays**, containing numbers (number 2D matrix). The input numbers will **always be positive**.

The **output** is a Boolean result indicating whether the matrix is magical or not.

Examples

Input	Output

`[[4, 5, 6],`
`[6, 5, 4],`
`[5, 5, 5]]`
	`true`

Input	Output

`[[11, 32, 45],`
`[21, 0, 1],`
`[21, 1, 1]]`
	`false`

Input	Output

`[[1, 0, 0],`
`[0, 0, 1],`
`[0, 1, 0]]`
	`true`

What to submit?

Function Signature: function main(matrix)

3. Tic-Tac-Toe

Make a tic-tac-toe console application.

You will receive an array of arrays. As you know there are two players in this game, so the first element of the input will be first player's chosen coordinates, the second element will be the second player's turn coordinates and so on.

The initial state of the dashboard is

```
[[false, false, false],
[false, false, false],
[false, false, false]]
```

The first player's mark is X and the second player's mark is O.

Input

One parameter:

- An array - **the moves** in row that players make

Output

- There are two players - X and O
- If a player tries to take his turn on a space that's already taken, he should make his turn again and you should print the following message:

  `"This space is already taken. Please choose another!"`

- If there are no free spaces on the dashboard and nobody wins, the game should end and you should print the following message:

  `"The game has ended! Nobody wins :("`

- If someone wins, you should print the following message and the **current state** of the **dashboard**:

  `"Player {X/0} wins"`

  Note: When printing the state of the dashboard, the elements of each row of the dashboard should be separated by **`"\t"`** and each row should be on new line.

Constraints

The elements in the input array will always be enough to end the game.

Examples

Input	Output
["0 1", "0 0", "0 2", "2 0", "1 0", "1 1", "1 2", "2 2", "2 1", "0 0"]	Player O wins! O X X X O X O false O
["0 0", "0 0", "1 1", "0 1", "1 2", "0 2", "2 2", "1 2", "2 2", "2 1"]	This space is already taken. Please choose another! Player X wins! X X X false O O false false false
["0 1", "0 0", "0 2", "2 0", "1 0", "1 2", "1 1", "2 1", "2 2", "0 0"]	The game has ended! Nobody wins :(O X X X X O O O X

What to submit?

Function Signature: function main(moves)

4. Diagonal Attack

Write a JS function that reads a given matrix of numbers and checks if both **main diagonals** have an **equal sum**. If they do, set every element that is **NOT** part of **the main diagonals** to that sum, alternatively, just print the matrix unchanged.

The **input** comes as an **array of strings**. Each element represents a **string of numbers** with **spaces** between them. Parse it into a **matrix of numbers** so you can work with it.

The **output** is either the new matrix with all cells not belonging to a main diagonal changed to the diagonal sum or the original matrix, if the two diagonals have different sums. You need to print **every row on a new line** with cells **separated by a space**. Check the examples below.

Examples

Input	Output
['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']	5 15 15 15 1 15 4 15 2 15 15 15 3 15 15 15 4 15 2 15 5 15 15 15 1

Input	Output
['1 1 1', '1 1 1' '1 1 0']	1 1 1 1 1 1 1 1 0

What to submit?

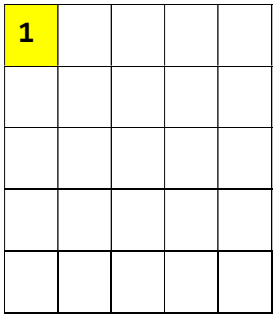
Function Signature: function main(strings)

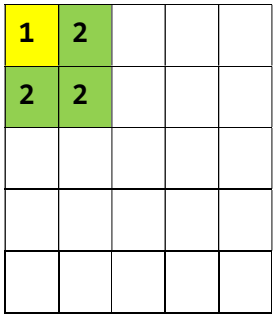
5. Orbit

You will be given an empty rectangular space of cells. Then you will be given the position of a star. You need to build the orbits around it.

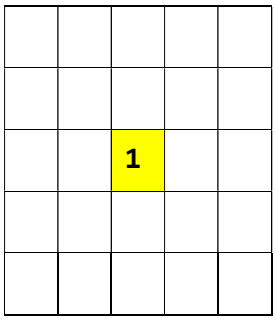
You will be given a coordinate of a cell, which will **always be inside the matrix**, on which you will put the value - **1**. Then you must set the values of the cells **directly surrounding that cell**, including the **diagonals**, **to 2**. After which you must set the values of the next surrounding cells to 3 and so on. Check the pictures for more information.

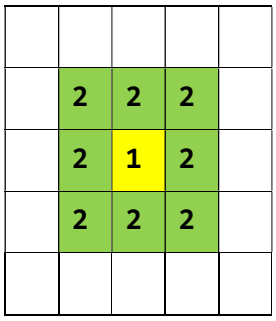
For example, we are given a matrix which has 5 rows and 5 columns, and the star is at coordinates - **0, 0**. Then the following should happen:

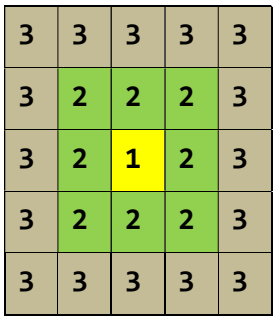




If the coordinates of the star are somewhere in the middle of the matrix for example - **2, 2**, then it should look like this:







The **input** comes as an **array of 4 numbers [width, height, x, y]** which represents the **dimensions of the matrix** and the **coordinates of the star.**

The **output** is the filled matrix, with the cells **separated by a space**, each **row on a new line**.

Examples

Input	Output

`[4, 4, 0, 0]`	`1 2 3 4`
	`2 2 3 4`
	`3 3 3 4`
	`4 4 4 4`

Input	Output

`[5, 5, 2, 2]`	`3 3 3 3 3`
	`3 2 2 2 3`
	`3 2 1 2 3`
	`3 2 2 2 3`
	`3 3 3 3 3`

Input	Output

`[3, 3, 2, 2]`	`3 3 3`
	`3 2 2`
	`3 2 1`

Hints

- Check if there is some **dependency** or **relation** between the **position of the numbers** and the **rows** and **columns** of those positions.

What to submit?

Function Signature: function main(dimensions)

6. Spiral Matrix

Write a JS function that generates a **spirally filled** matrix with numbers, with given dimensions.

The **input** comes as **2 numbers** that represent the **dimension of the matrix**.

The **output** is the matrix filled spirally starting from **1**. You need to print **every row on a new line** with the cells **separated by a space**. Check the examples below.

Examples

Input	Output

`5, 5`	`1 2 3 4 5`
	`16 17 18 19 6`
	`15 24 25 20 7`
	`14 23 22 21 8`
	`13 12 11 10 9`

Input	Output

`3, 3`	`1 2 3`
	`8 9 4`
	`7 6 5`

What to submit?

Function Signature: function main(input1, input2)