# Exercises: Arrays

Problems for exercises and homework for the ["Technology Fundamentals" course @ SoftUni](https://softuni.bg/modules/57/tech-module-4-0).

You can check your solutions here: [Arrays-Exercise](https://judge.softuni.bg/Contests/1256/Arrays-Exercise)

## Add or Subtract

Write a JS function, which changes the **value** of **odd** and **even** numbers in an **array of numbers**. If the number is **even** **add** to its value its **index** position. If the number is **odd** **subtract** to its value its index position. On the first line print the **newly modified array**, on the second line print the **sum** of numbers from the **original array**, on the third line print the sum of numbers from the **modified array**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| [5, 15, 23, 56, 35] | [ 5, 14, 21, 59, 31 ]  134  130 |
| [-5, 11, 3, 0, 2] | [ -5, 10, 1, 3, 6 ]  11  15 |

## Common Elements

Write a JS function, which prints common elements in **two** arrays. Print all matches on **separate** lines. Compare the **first array** with the **second array**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| ["Hey", "hello", 2, 4, "Pesho", "e"], ["Pecho", 10, "hey", 4, "hello", "2"] | hello  4 |
| ["S", "o", "f", "t", "U", "n", "i", " "],  ["s", "o", "c", "i", "a", "l"] | o  i |

## Merge Arrays

Write a JS function which receives **two arrays** and **merges** them into a third array. If the **index** of the element is **even**, add into the third array the **sum** of **both elements** at that index. If the index of the element is **odd**, add the **concatenation** of both elements at that index.

As **input** you will receive **two string arrays**.

As **output** you should print the resulting **third** array, each element separated by **" - "**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| ["5", "15", "23", "56", "35"],  ["17", "22", "87", "36", "11"] | 22 - 1522 - 110 - 5636 - 46 |
| ["13", "12312", "5", "77", "4"],  ["22", "333", "5", "122", "44"] | 35 - 12312333 - 10 - 77122 - 48 |

## Array Rotation

Write a JS function that receives an **array** and **number of rotations** you have to perform (first element goes at the end) Print the resulting array.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| [51, 47, 32, 61, 21],  2 | 32 61 21 51 47 |
| [32, 21, 61, 1],  4 | 32 21 61 1 |
| [2, 4, 15, 31],  5 | 4 15 31 2 |

## Max Number

Write a JS function to find all the top integers in an array. A top integer is an integer which is **bigger** than all the elements to its right. Print **all** top integers on the console.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| [1, 4, 3, 2] | 4 3 2 |
| [14, 24, 3, 19, 15, 17] | 24 19 17 |
| [41, 41, 34, 20] | 41 34 20 |
| [27, 19, 42, 2, 13, 45, 48] | 48 |

## Equal Sums

Write a JS function that determines if there **exists an element in the array** such that the **sum of the elements on its left** is **equal** to the **sum of the elements on its right**. If there are **no elements to the left / right**, their **sum is considered to be 0**. Print the **index** that satisfies the required condition or **"no"** if there is no such index.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| [1, 2, 3, 3] | 2 | At a[2] -> left sum = 3, right sum = 3  a[0] + a[1] = a[3] |
| [1, 2] | no | At a[0] -> left sum = 0, right sum = 2  At a[1] -> left sum = 1, right sum = 0  No such index exists |
| [1] | 0 | At a[0] -> left sum = 0, right sum = 0 |
| [1, 2, 3] | no | No such index exists |
| [10, 5, 5, 99, 3, 4, 2, 5, 1, 1, 4] | 3 | At a[3] -> left sum = 20, right sum = 20  a[0] + a[1] + a[2] = a[4] + a[5] + a[6] + a[7] + a[8] + a[9] + a[10] |

## Max Sequence of Equal Elements

Write a JS function that finds the **longest sequence of equal elements** in an array of numbers. If several longest sequences exist, print the leftmost one.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| ['2 1 1 2 3 3 2 2 2 1'] | 2 2 2 |
| ['1 1 1 2 3 1 3 3'] | 1 1 1 |
| ['4 4 4 4'] | 4 4 4 4 |
| ['0 1 1 5 2 2 6 3 3'] | 1 1 |

## Magic Sum

Write a JS function, which prints all unique pairs in an array of integers whose sum is equal to a given number.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| ['1 7 6 2 19 23', '8'] | 1 7  6 2 |
| ['14 20 60 13 7 19 8', '27'] | 14 13  20 7  19 8 |
| ['1 2 3 4 5 6', '6'] | 1 5  2 4 |

## Tseam Account

As a gamer, Pesho has Tseam Account. He loves to buy new games. You are given Pesho's account with all of his games-> strings, separated by space. Until you receive **"Play!"** you will be receiving commands which Pesho does with his account.

You may receive the following commands:

* **Install {game}**
* **Uninstall {game}**
* **Update {game}**
* **Expansion {game}-{expansion}**

If you receive **Install command**, you should **add** the game at last position in the account, but only if it isn't installed already.

If you receive **Uninstall command**, **delete** the game if it exists.

If you receive **Update command**, you should **update** the game if it exists and place it on **last position**.

If you receive **Expansion command**, you should check if the game exists and **insert** after it the expansion in the following format: "**{game}:{expansion}";**

**Input**

* On the **first input line** you will receive Pesho`s **account** – sequence of game names, separated by space.
* Until you receive **"Play!"** you will be receiving **commands**.

**Output**

* As output you must print Pesho`s Tseam **account**.

**Constraints**

* The **command will always be valid.**
* The **game** and **expansion** will be strings and will contain any character, except **'-'**.
* Allowed working **time** / **memory**: **100ms** / **16MB**.

**Examples**

|  |  |  |
| --- | --- | --- |
| Input | Output | Comment |
| CS WoW Diablo  Install LoL  Uninstall WoW  Update Diablo  Expansion CS-Go  Play! | CS CS:Go LoL Diablo | We receive the account => CS, WoW, Diablo  We Install LoL => CS, WoW, Diablo, LoL  Uninstall WoW => CS, Diablo, LoL  Update Diablo => CS, LoL, Diablo  We add expansion => CS, CS:Go, LoL, Diablo  We print the account. |
| CS WoW Diablo  Uninstall XCOM  Update PeshoGame  Update WoW  Expansion Civ-V  Play! | CS Diablo WoW |  |

## 10 \*Ladybugs

You are **given a field size** and the **indexes of ladybugs** inside the field. A **ladybug changes its position** either to its **left or to its right** **by a given fly length**. A **command to a ladybug** looks like this: "**0 right 1**". This means that the little insect placed on index 0 should fly one index to its right. If the ladybug **lands on a fellow ladybug**, it **continues to fly** in the same direction **by the same fly length**. If the ladybug **flies out of the field, it is gone**.

For example, imagine you are given a field with size 3 and ladybugs on indexes 0 and 1. If the ladybug on index 0 needs to fly to its right by the length of 1 (0 right 1) it will attempt to land on index 1 but as there is another ladybug there it will continue further to the right by additional length of 1, landing on index 2. After that, if the same ladybug needs to fly to its right by the length of 1 (2 right 1), it will land somewhere outside of the field, so it flies away:



If you are given ladybug index that does not have ladybug there, nothing happens. If you are given ladybug index that is outside the field, nothing happens.

Your job is to create the program, simulating the ladybugs flying around doing nothing. At the end, **print all cells in the field separated by blank spaces**. For each cell that has a ladybug on it print '**1**' and for each empty cells print '**0**'. For the example above, the output should be **'0 1 0'**.

### Input

* You will receive an **array of strings** and the first element is an **integer** – the **size** of the field
* The second element is a **string** containing the initial **indexes** of all ladybugs separated by a blank space. **The given indexes** may or may not be inside the field range
* The next elements in the **array** are commands in the format: "**{ladybug index} {direction} {fly length}**"

### Output

* Print the **all cells within the field in format: "{cell} {cell} … {cell}"**
  + If a cell has ladybug in it, print **'1'**
  + If a cell is empty, print **'0'**

### Constrains

* The size of the field will be in the range [0 … 1000]
* The ladybug indexes will be in the range [-2,147,483,647… 2,147,483,647]
* The number of commands will be in the range [0 … 100]
* The fly length will be in the range [-2,147,483,647… 2,147,483,647]

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| [ 3, '0 1',  '0 right 1',  '2 right 1' ] | 0 1 0 | 1 1 0 - Initial field  0 1 1 - field after "0 right 1"  0 1 0 - field after "2 right 1" |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| [ 3, '0 1 2',  '0 right 1',  '1 right 1',  '2 right 1'] | 0 0 0 |  | [ 5, '3',  '3 left 2',  '1 left -2'] | 0 0 0 1 0 |