Exercise: JS Fundamentals

Exercise problems for the "Back-End Technologies Basics" Course @ SoftUni.
You can check your solutions in Judge.

1. Array Rotation

Write a function that receives an **array** and the **number of rotations** you have to perform.

Note: Depending on the number of rotations, the first element goes to the end.

Output

Print the resulting array elements separated by a single space.

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`

2. Print Every N-th Element from an Array

The **input** comes as two parameters – an **array of strings** and a **number**. The second parameter is **N – the step**.

The **output** is every element on the **N-th** step **starting from the first one**. If the step is **3**, you need to return the **1st**, the **4th**, the **7th** … and so on, until you reach the end of the array.

The **output** is the **return** value of your function and must be an **array**.

Example

Input	Output
`['5', '20', '31', '4', '20'], 2`	`['5', '31', '20']`

Input	Output
`['dsa', 'asd', 'test', 'tset'], 2`	`['dsa', 'test']`

Input	Output
`['1', '2', '3', '4', '5'], 6`	`['1']`

Hints

Return all the elements with **for** loop, **incrementing** the **loop variable** with the value of the **step** variable.

3. List of Names

You will receive an **array of names**. Sort them **alphabetically in ascending order** and print a numbered list of all the names, each on a new line.

Example

Input	Output
["John", "Bob", "Christina", "Ema"]	1.Bob 2.Christina 3.Ema 4.John

Hints

The **sort function** rearranges the array in ascending order

4. Sorting Numbers

Write a function that sorts an **array of numbers** so that the first element is the **smallest** one, the second is the **biggest** one, the third is the **second smallest** one, the fourth is the **second biggest** one, and so on.

Return the resulting array.

Example

Input	Output
[1, 65, 3, 52, 48, 63, 31, -3, 18, 56]	[-3, 65, 1, 63, 3, 56, 18, 52, 31, 48]

5. Reveal Words

Write a function, which receives **two parameters**.

The first parameter will be a string with some words **separated by ', '**.

The second parameter will be a string that contains **templates containing '*'**.

Find the word with the **same length** as the template and **replace** it.

Example

Input	Output
'great', 'softuni is ***** place for learning new programming languages'	softuni is great place for learning new programming languages
'great, learning', 'softuni is ***** place for ******** new programming languages'	softuni is great place for learning new programming languages

6. String Substring

The input will be given as **two** separated strings (a **word** as a first parameter and a **text** as a second).

Write a function that checks given text for containing a given word. The comparison should be **case insensitive.** Once you find a match, **print** the word and **stop** the program.

If you don't find the word print: **"{word} not found!"**

Example

Input	Output

`'javascript',`
`'JavaScript is the best programming language'`
	`javascript`

`'python',`
`'JavaScript is the best programming language'`
	`python not found!`

7. Smallest of Three Numbers

Write a function that receives **three integers** and prints the **smallest** number. Use an appropriate name for the function.

Examples

Input	Output

`2,`
`5,`
`3`
	`2`

`600,`
`342,`
`123`
	`123`

`25,`
`21,`
`4`
	`4`

`2,`
`2,`
`2`
	`2`

8. Add and Subtract

You will receive **three integer numbers.**

Write a function **`sum()`** to calculate the sum of the first **two** integers and a function **`subtract()`**, which subtracts the result of the function the **`sum()`** and the **third** integer.

Examples

Input	Output

`23,`
`6,`
`10`
	`19`

`1,`	`-12`

17, 30	
42, 58, 100	0

9. Odd and Even Sum

You will receive a **single number.** You have to write a function, that returns the **sum** of **all even** and **all odd** digits from that number.

Examples

Input	Output
1000435	Odd sum = 9, Even sum = 4
3495892137259234	Odd sum = 54, Even sum = 22

10. Password Validator

Write a function that checks if a given password is valid. Password validations are:

- The **length** should be **6 - 10** characters (inclusive)
- It should consist **only of letters** and **digits**
- It should have **at least 2** digits

If a password is a valid print: **"Password is valid"**.

If it is **NOT** valid, for every unfulfilled rule print a message:

- **"Password must be between 6 and 10 characters"**
- **"Password must consist only of letters and digits"**
- **"Password must have at least 2 digits"**

Examples

Input	Output
'logIn'	Password must be between 6 and 10 characters Password must have at least 2 digits
'MyPass123'	Password is valid
'Pa$s$s'	Password must consist only of letters and digits Password must have at least 2 digits

11. Employees

You're tasked to create a list of employees and their personal numbers.

You will receive an array of strings. Each string is an employee **name** and to assign them a personal number you have to find the **length of the name** (whitespace included).

Try to use an object.

At the end print all the list employees in the following format:

`"Name: {employeeName} -- Personal Number: {personalNum}"`

Examples

Input	Output

`[`
`'Silas Butler',`
`'Adnaan Buckley',`
`'Juan Peterson',`
`'Brendan Villarreal'`
`]`
	`Name: Silas Butler -- Personal Number: 12`
	`Name: Adnaan Buckley -- Personal Number: 14`
	`Name: Juan Peterson -- Personal Number: 13`
	`Name: Brendan Villarreal -- Personal Number: 18`

`[`
`'Samuel Jackson',`
`'Will Smith',`
`'Bruce Willis',`
`'Tom Holland'`
`]`
	`Name: Samuel Jackson -- Personal Number: 14`
	`Name: Will Smith -- Personal Number: 10`
	`Name: Bruce Willis -- Personal Number: 12`
	`Name: Tom Holland -- Personal Number: 11`

12. Towns

You're tasked to create and print **objects** from a text table.

You will receive the input as an **`array`** of strings, where each string represents a table row, with values on the row separated by pipes " **|** " and spaces.

The table will consist of exactly 3 columns **`"Town"`**, **`"Latitude"`** and **`"Longitude"`**. The **`latitude`** and **`longitude`** columns will always contain **valid numbers**. Check the examples to get a better understanding of your task.

The **output** should be **objects**. Latitude and longitude must be parsed to **numbers and formatted to the second decimal point**!

Examples

Input

`['Sofia | 42.696552 | 23.32601',`
`'Beijing | 39.913818 | 116.363625']`

Output

`{ town: 'Sofia', latitude: '42.70', longitude: '23.33' }`
`{ town: 'Beijing', latitude: '39.91', longitude: '116.36' }`

Input
`['Plovdiv | 136.45 | 812.575']`
Output
`{ town: 'Plovdiv', latitude: '136.45', longitude: '812.58' }`

13. Store Provision

You will receive **two arrays**. The first array represents the current **stock** of the local store. The second array will contain **products** that the store has **ordered** for delivery.

The following information applies to both arrays:

Every **even** index will hold the **name** of the **`product`** and every **odd** index will hold the **`quantity`** of that **`product`**. The second array could contain products that are **already in** the local store. If that happens **increase** the **quantity** for the given product. You should store them into an **`object`**, and print them in the following format: **`(product -> quantity)`**

All of the arrays' values will be **strings.**

Examples

Input	Output

`[`
`'Chips', '5', 'CocaCola', '9', 'Bananas', '14', 'Pasta', '4', 'Beer', '2'`
`],`
`[`
`'Flour', '44', 'Oil', '12', 'Pasta', '7', 'Tomatoes', '70', 'Bananas', '30'`
`]`
	`Chips -> 5`
	`CocaCola -> 9`
	`Bananas -> 44`
	`Pasta -> 11`
	`Beer -> 2`
	`Flour -> 44`
	`Oil -> 12`
	`Tomatoes -> 70`

`[`
`'Salt', '2', 'Fanta', '4', 'Apple', '14', 'Water', '4', 'Juice', '5'`
`],`
`[`
`'Sugar', '44', 'Oil', '12', 'Apple', '7', 'Tomatoes', '7', 'Bananas', '30'`
`]`
	`Salt -> 2`
	`Fanta -> 4`
	`Apple -> 21`
	`Water -> 4`
	`Juice -> 5`
	`Sugar -> 44`
	`Oil -> 12`
	`Tomatoes -> 7`
	`Bananas -> 30`

14. Movies

Write a function that stores information about movies inside an array. The movie's object info must be **`name`, `director`,** and **`date`**. You can receive several types of input:

- **`"addMovie {movie name}"`** – add the movie

- **`"{movie name} directedBy {director}"`** – check if the movie **exists** and then add the director
- **`"{movie name} onDate {date}"`** – check if the movie **exists** and then add the date

At the end print all the movies that have **all the info** (if the movie has **no** director, name, or date, **don't** print it) in **JSON format.**

Examples

Input	Output

[
'addMovie Fast and Furious',
'addMovie Godfather',
'Inception directedBy Christopher Nolan',
'Godfather directedBy Francis Ford Coppola',
'Godfather onDate 29.07.2018',
'Fast and Furious onDate 30.07.2018',
'Batman onDate 01.08.2018',
'Fast and Furious directedBy Rob Cohen'
]
	{"name":"Fast and Furious","date":"30.07.2018","director":"Rob Cohen"}
	{"name":"Godfather","director":"Francis Ford Coppola","date":"29.07.2018"}

['addMovie The Avengers', 'addMovie Superman', 'The Avengers directedBy Anthony Russo', 'The Avengers onDate 30.07.2010', 'Captain America onDate 30.07.2010', 'Captain America directedBy Joe Russo']	{"name":"The Avengers","director":"Anthony Russo","date":"30.07.2010"}

15. Inventory

Create a function, which creates a **register for heroes**, with their **names**, **level**, and **items** (if they have such).

The **input** comes as an **array of strings**. Each element holds data for a hero, in the following format:

`"{heroName} / {heroLevel} / {item1}, {item2}, {item3}..."`

You must store the data about every hero. The **name** is a **string**, a **level** is a **number** and the items are all **strings.**

The **output** is all of the data for all the heroes you've stored **sorted ascending by level**. The data must be in the following format for each hero:

```
Hero: {heroName}

level => {heroLevel}

Items => {item1}, {item2}, {item3}
```

Examples

Input	Output

`[`
`'Isacc / 25 / Apple, GravityGun',`
`'Derek / 12 / BarrelVest, DestructionSword',`
`'Hes / 1 / Desolator, Sentinel, Antara'`
`]`
	`Hero: Hes`
	`level => 1`
	`items => Desolator, Sentinel, Antara`
	`Hero: Derek`
	`level => 12`
	`items => BarrelVest, DestructionSword`
	`Hero: Isacc`
	`level => 25`
	`items => Apple, GravityGun`

`[`
`'Batman / 2 / Banana, Gun',`
`'Superman / 18 / Sword',`
`'Poppy / 28 / Sentinel, Antara'`
`]`
	`Hero: Batman`
	`level => 2`
	`items => Banana, Gun`
	`Hero: Superman`
	`level => 18`
	`items => Sword`
	`Hero: Poppy`
	`level => 28`
	`items => Sentinel, Antara`

16. Odd Occurrences

Write a function that extracts the elements of a sentence, if it appears an odd number of times (**case-insensitive**).

The input comes as a **single string**. The words will be **separated by a single space**.

Example

Input	Output
`'Java C# Php PHP Java PhP 3 C# 3 1 5 C#'`	`c# php 1 5`
`'Cake IS SWEET is Soft CAKE sweet Food'`	`soft food`

17. Piccolo

Write a function that:

- Records a **car number** for every car that enters the **parking lot**
- Removes a **car number** when the car goes out
- Input will be an array of strings in format **`[direction, carNumber]`**

Print the output with all car numbers which are in the parking lot **sorted in ascending by number.**

If the parking lot is empty, print: **"`Parking Lot is Empty`"**.

Examples

Input	Output

`['IN, CA2844AA',`
`'IN, CA1234TA',`
`'OUT, CA2844AA',`
`'IN, CA9999TT',`
`'IN, CA2866HI',`
`'OUT, CA1234TA',`
`'IN, CA2844AA',`
`'OUT, CA2866HI',`
`'IN, CA9876HH',`
`'IN, CA2822UU']`
	`CA2822UU`
	`CA2844AA`
	`CA9876HH`
	`CA9999TT`

`['IN, CA2844AA',`
`'IN, CA1234TA',`
`'OUT, CA2844AA',`
`'OUT, CA1234TA']`
	`Parking Lot is Empty`