Exercise: Objects and Classes

Problems for exercise and homework for the "JS Fundamentals" Course @ SoftUni.
Submit your solutions in the SoftUni judge system at: https://judge.softuni.org/Contests/1322

1. 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

2. 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' }

3. 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

4. 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"}

5. 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

6. Make a Dictionary

You will receive an **array** with **strings in the form of JSON's.**

You have to parse these strings and combine them into **one object**. Every string from the array will hold **terms** and a **description.** If you receive the **same term twice,** replace it with the **new definition**.

Print every term and definition in that dictionary on new line in format:

`Term: ${term} => Definition: ${definition}`

Don't forget to sort the dictionary **alphabetically** by the terms as in real dictionaries.

Examples

Input	Output
['{"Coffee":"A hot drink made from the roasted and ground seeds (coffee beans) of a tropical shrub."}', '{"Bus":"A large motor vehicle carrying passengers by road, typically one serving the public on a fixed route and for a fare."}', '{"Boiler":"A fuel-burning apparatus or container for heating water."}', '{"Tape":"A narrow strip of material, typically used to hold or fasten something."}', '{"Microphone":"An instrument for converting sound waves into electrical energy variations which may then be amplified, transmitted, or recorded."}']	Term: Boiler => Definition: A fuel-burning apparatus or container for heating water. Term: Bus => Definition: A large motor vehicle carrying passengers by road, typically one serving the public on a fixed route and for a fare. Term: Coffee => Definition: A hot drink made from the roasted and ground seeds (coffee beans) of a tropical shrub. Term: Microphone => Definition: An instrument for converting sound waves into electrical energy variations which may then be amplified, transmitted, or recorded. Term: Tape => Definition: A narrow strip of material, typically used to hold or fasten something.
['{"Cup":"A small bowl-shaped container for drinking from, typically having a handle"}', '{"Cake":"An item of soft sweet food made from a mixture of flour, fat, eggs, sugar, and other ingredients, baked and sometimes iced or decorated."} ', '{"Watermelon":"The large fruit of a plant of the gourd family, with smooth green skin, red pulp, and watery juice."} ', '{"Music":"Vocal or instrumental sounds (or both) combined in such a way as to produce beauty of form, harmony, and expression of emotion."} ', '{"Art":"The expression or application of human creative skill and imagination, typically in a visual form such as	Term: Art => Definition: The expression or application of human creative skill and imagination, typically in a visual form such as painting or sculpture, producing works to be appreciated primarily for their beauty or emotional power. Term: Cake => Definition: An item of soft sweet food made from a mixture of flour, fat, eggs, sugar, and other ingredients, baked and sometimes iced or decorated. Term: Cup => Definition: A small bowl-shaped container for drinking from, typically having a handle Term: Music => Definition: Vocal or instrumental sounds (or both) combined in such a way as to produce beauty of form, harmony, and expression of emotion. Term: Watermelon => Definition: The large fruit of a plant of the gourd

painting or sculpture, producing works to be appreciated primarily for their beauty or emotional power."} ']	family, with smooth green skin, red pulp, and watery juice.

7. Class Vehicle

Create a class with the name **Vehicle** that has the following properties:

- **type** – a string
- **model** – a string
- **parts** – an object that contains:
  - **engine** – number (quality of the engine)
  - **power** – number
  - **quality** – engine * power
- **fuel** – a number
- **drive** – a function that receives fuel loss and decreases the fuel of the vehicle by that number

The **constructor** should receive the **type**, the **model**, the **parts** as an **object,** and the **fuel**

In judge post your **class** (**Note: all names should be as described**)

Example

Test your Vehicle class.

Input	Output

`let parts = { engine: 6, power: 100 };`
`let vehicle = new Vehicle('a', 'b', parts, 200);`
`vehicle.drive(100);`
`console.log(vehicle.fuel);`
`console.log(vehicle.parts.quality);`
	100
	600

`let parts = {engine: 9, power: 500};`
`let vehicle = new Vehicle('l', 'k', parts, 840);`
`vehicle.drive(20);`
`console.log(vehicle.fuel);`
	820

8. *Class Storage

Create a **class Storage**. It should have the following **properties**, while the **constructor** should only receive a **capacity**:

- **capacity** – a number that **decreases when adding a given quantity** of products to storage
- **storage** – **list of products** (object). **Each product** should have:
  - **name** - a string
  - **price** – a number (price is for a single piece of product)
  - **quantity** – a number
- **totalCost** – the sum of the cost of the products

The class should also have the following **methods**:

- **addProduct** – a function that receives a product and adds it to the storage
- **getProcuts** – a function that returns all the products in storage in **JSON** format, each on a new line

Paste only the **class Storage in judge** (**Note: all names should be as described**)

Example

Test your Storage class.

Input	Output

`let productOne = {name: 'Cucamber', price: 1.50, quantity: 15};`
`let productTwo = {name: 'Tomato', price: 0.90, quantity: 25};`
`let productThree = {name: 'Bread', price: 1.10, quantity: 8};`
`let storage = new Storage(50);`
`storage.addProduct(productOne);`
`storage.addProduct(productTwo);`
`storage.addProduct(productThree);`
`console.log(storage.getProducts());`
`console.log(storage.capacity);`
`console.log(storage.totalCost);`
	`{"name":"Cucamber","price":1.5,"quantity":15}`
	`{"name":"Tomato","price":0.9,"quantity":25}`
	`{"name":"Bread","price":1.1,"quantity":8}`
	`2`
	`53.8`

`let productOne = {name: 'Tomato', price: 0.90, quantity: 19};`
`let productTwo = {name: 'Potato', price: 1.10, quantity: 10};`
`let storage = new Storage(30);`
`storage.addProduct(productOne);`
`storage.addProduct(productTwo);`
`console.log(storage.totalCost);`
	`28.1`

9. *Catalogue

You have to create a sorted catalog of store **products**. You will be given the products' **names** and **prices**. You need to order them in **alphabetical order**.

The **input** comes as an **array** of strings. Each element holds info about a product in the following format:

```
"{productName} : {productPrice}"
```

The **product's name** will be a **string**, which will **always start with a capital letter**, and the **price** will be **a number**. You can safely assume there will be **NO duplicate product input**. The comparison for alphabetical order is **case-insensitive**.

As **output,** you must print all the products in a specified format. They must be ordered **exactly as specified above**. The products must be **divided into groups**, by the **initial of their name**. The **group's initial should be printed**, and after that, the products should be printed with **2 spaces before their names**. For more info check the examples.

Examples

Input	Output

[
'Appricot : 20.4',
'Fridge : 1500',
'TV : 1499',
'Deodorant : 10',
'Boiler : 300',
'Apple : 1.25',
'Anti-Bug Spray : 15',
'T-Shirt : 10'
]
	A
	  Anti-Bug Spray: 15
	  Apple: 1.25
	  Appricot: 20.4
	B
	  Boiler: 300
	D
	  Deodorant: 10
	F
	  Fridge: 1500
	T
	  T-Shirt: 10
	  TV: 1499

[
'Omlet : 5.4',
'Shirt : 15',
'Cake : 59'
]
	C
	  Cake: 59
	O
	  Omlet: 5.4
	S
	  Shirt: 15