Exercises: Objects

1. Heroic Inventory

Create a function that **register heroes** from an array of strings. Each string in the array has a format and it contains the hero **name, level** and **may or may not contain items**. You must extract the data and store it in an array. Print the result as JSON.

Input

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

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

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

Output

The **output** is a **JSON representation** of the data for all the heroes you've stored. The data must be an **array of all the heroes**. Check the examples for more info.

Examples

Input	Output
['Isacc / 25 / Apple, GravityGun', 'Derek / 12 / BarrelVest, DestructionSword', 'Hes / 1 / Desolator, Sentinel, Antara']	[{"name":"Isacc","level":25,"items":["Apple","GravityGun"]}, {"name":"Derek","level":12,"items":["BarrelVest","DestructionSword"]},{"name":"Hes","level":1,"items":["Desolator","Sentinel","Antara"]}]
['Jake / 1000 / Gauss, HolidayGrenade']	[{"name":"Jake","level":1000,"items":["Gauss","HolidayGrenade"]}]

Hints

- We need an array that will hold our hero data. That is the first thing we create.

```
function main(input) {
    let heroData = [
    ];
}
```

- Next, we need to loop over the whole input, and process it. Let's do that using a simple **for** loop.

```
function main(input) {
    let heroData = [
    ];

    for(let i = 0; i < input.length; i++) {
        let currentHeroArguments = input[i].split(" / ");
    }
}
```

- Every element from the input holds data about a hero. However, the **elements from the data** we need are **separated by some delimiter**, so we just split each string with that **delimiter**.

- Next, we need to take the elements from the **string array**, which is a result of the **string split**, and parse them.

```
for(let i = 0; i < input.length; i++) {
    let currentHeroArguments = input[i].split(" / ");

    let currentHeroName = currentHeroArguments[0];
    let currentHeroLevel = Number(currentHeroArguments[1]);
    let currentHeroItems = currentHeroArguments[2].split(", ");
}
```

- However, if you do this, you could get an error. If you go up and read the problem definition again, there might be a **case** where a hero **has no items.** In that case, if we try to take the **3rd element** of the **`currentHeroArguments`** array, it will **result in an error**. That is why we should perform a simple check.

```
let currentHeroItems = [];

if(currentHeroArguments.length > 2) {
    currentHeroItems = currentHeroArguments[2].split(", ");
}
```

- If **there are any items** in the **input**, we should **split them**. If there's not, it will just remain an **empty array**.
- After we extracted the data needed, we can now store them in an **object** and **add** it to the **array**.

```
let hero = {
    name: currentHeroName,
    level: currentHeroLevel,
    items: currentHeroItems
};

heroData.push(hero);
```

- Finally, we need to turn the array of objects we have made into a JSON string, which can be done by using **`JSON.stringify()`** function.

```
console.log(JSON.stringify(heroData));
```

What to submit?

Function Signature: function main(input)

2. JSON's Table

JSON's Table is a magical table that turns JSON data into an HTML table. You will be given **JSON strings** that hold the data about the employees, including their **name**, **position**, and **salary**. You need to **parse that data** into **objects** and create an **HTML table** that holds the data for each **employee on different rows**.

The **name** and **position** of the employee are **strings** and a **salary** is a **number**.

Input

The **input** comes as an array of strings. Each element is a JSON string represents the data for each employee.

Output

The **output** is an HTML code of a table that holds the data of all the employees. Check the examples for more info.

Examples

Input	Output
['{"name":"Peter","position":"Manager","salary":100000}', '{"name":"Teo","position":"Lecturer","salary":1000}', '{"name":"George","position":"Lecturer","salary":1000}']	<table> <tr> <td>Peter</td> <td>Manager</td> <td>100000</td> </tr> <tr> <td>Teo</td> <td>Lecturer</td> <td>1000</td> </tr> <tr> <td>George</td> <td>Lecturer</td> <td>1000</td> </tr> </table>

Hints

- You might want to **escape the HTML**. Otherwise, you might find yourself a victim to vicious JavaScript **code in the input**.

What to submit?

Function Signature: function main(employees)

3. Cappy Juice

The input comes as an **array of strings** and each element contains **juice** and its **quantity**. Count the **number of bottles** that every juice can produce and print them.

Note: Multiple elements may contain the **same juice** and their quantity must be **combined**. Every **1000 quantity** of juice is **one bottle**. Any excess or remainder after producing a bottle should be stored to use it later when another data with the same juice is received.

Example: You have 2643 quantity of Orange Juice – this is **2 bottles** of Orange Juice and **643 quantity left**.

Input

The **input** comes as array of strings. Each element holds the data about a juice and quantity in the following format:

"**`{juiceName} => {juiceQuantity}`**"

Output

The **output** is the produced bottles. The bottles are to be printed in **order of obtaining the bottles**. Check the second example bellow - even though we receive the Kiwi juice first, we can't form a bottle of Kiwi juice until the 4th line, at which point we have already created Pear and Watermelon juice bottles; thus, the Kiwi bottles appear last in the output.

Examples

Input	Output
['Orange => 2000', 'Peach => 1432', 'Banana => 450', 'Peach => 600', 'Strawberry => 549']	Orange => 2 Peach => 2
['Kiwi => 234', 'Pear => 2345', 'Watermelon => 3456', 'Kiwi => 4567', 'Pear => 5678', 'Watermelon => 6789']	Pear => 8 Watermelon => 10 Kiwi => 4

What to submit?

Function Signature: function main(juices)

4. Store Catalog

You have to create a sorted catalog of store products. You will be given an array of strings that contains the product's name and price. You need to group them by their initials and sort them in **alphabetical order**.

Input

The **input** comes as 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**.

Output

As **output**, you must print all the products in a specified format. You must group and order them **exactly as specified above**. The **group initials** should be printed, and their products should be printed below them with **2 spaces before their names**. For more info check the examples.

Examples

Input	Output
['Appricot : 20.4', 'Fridge : 1500',	A Anti-Bug Spray: 15

Input	Output
['Banana : 2', 'Rubic's Cube : 5',	B Banana: 2

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

'Raspberry P : 4999', 'Rolex : 100000', 'Rollon : 10', 'Rali Car : 2000000', 'Peter : 0.000001', 'Barrel : 10']	Barrel: 10 P Peter: 0.000001 R Rali Car: 2000000 Raspberry P: 4999 Rolex: 100000 Rollon: 10 Rubic's Cube: 5

What to submit?

Function Signature: function main(items)

5. Auto-Engineering Company

You are tasked to create a register for a company that produces cars. You need to store **how many cars** have been produced from a **specified model** of a **specified brand**.

Input

The **input** comes as array of strings. Each element holds information in the following format:

"**{carBrand} | {carModel} | {producedCars}**"

The car **brands** and **models** are **strings**, the **produced cars** are **numbers**. If the **car brand** you've received **already exists**, just add the **new car model** to it with the **produced cars as its value**. If even the car model exists, just **add** the **given value** to the **current one**.

Output

As **output,** you need to print - **for every car brand**, the **car models**, and **number of cars produced** from that model. The output format is:

"{carBrand}

###{carModel} -> {producedCars}

###{carModel2} -> {producedCars}

..."

The order of printing is the **order in which the brands and models first appeared from the input**. The first brand in the input should be the first printed and so on. For each brand, the first model received from that brand, should be the first printed and so on.

Examples

Input	Output
['Audi | Q7 | 1000', 'Audi | Q6 | 100', 'BMW | X5 | 1000', 'BMW | X6 | 100',	Audi ###Q7 -> 1000 ###Q6 -> 100 BMW

'Citroen | C4 | 123', 'Volga | GAZ-24 | 1000000', 'Lada | Niva | 1000000', 'Lada | Jigula | 1000000', 'Citroen | C4 | 22', 'Citroen | C5 | 10']	###X5 -> 1000 ###X6 -> 100 Citroen ###C4 -> 145 ###C5 -> 10 Volga ###GAZ-24 -> 1000000 Lada ###Niva -> 1000000 ###Jigula -> 1000000

What to submit?

Function Signature: function main(cars)