Exercise: Comprehensions

Problems for exercise and homework for the Python Advanced Course @SoftUni. Submit your solutions in the SoftUni judge system at https://judge.softuni.bg/Contests/1837

1. Words Lengths

Using **list comprehension**, write a program that receives some **strings** separated by comma and space **", "** and prints on the console each **string** with its **length** in the format: **"{first_str} -> {first_str_len}, {second_str} -> {second_str_len}…"**

Examples

Input	Output
`Peter, George, Bill, Lilly, Katy`	`Peter -> 5, George -> 6, Bill -> 4, Lilly -> 5, Katy -> 4`
`Some, Random, Text`	`Some -> 4, Random -> 6, Text -> 4`

2. Number Classification

Using **list comprehension** write a program that receives numbers separated by comma and space **", "** and prints all the **positive**, **negative**, **even** and **odd** numbers on separate lines as shown below.

Note: Zero is counted for a positive number

Examples

Input	Output

`1, -2, 0, 5, 3, 4, -100, -20, 12, 19, -33`	`Positive: 1, 0, 5, 3, 4, 12, 19`
	`Negative: -2, -100, -20, -33`
	`Even: -2, 0, 4, -100, -20, 12`
	`Odd: 1, 5, 3, 19, -33`

3. Diagonals

Using **nested list comprehension** write a program that reads **NxN** matrix, finds its **diagonals**, prints them and their **sum** as shown below.

Examples

Input	Output

`3`
`1, 2, 3`
`4, 5, 6`
`7, 8, 9`
	`First diagonal: 1, 5, 9. Sum: 15`
	`Second diagonal: 3, 5, 7. Sum: 15`

4. Capitals

Using **dictionary comprehension** write a program that receives **countries** on the first line separated by **comma and space ", "** and their corresponding **capital cities** on the second line (again separated by **comma and space ", "**) and **prints each country** with their **capital** on a **separate line** in the format: **"{country} -> {capital}"**

Hints

- You can use the **`zip()`** method to zip the two lists into **tuple pairs.**

Examples

Input	Output

`Bulgaria, Romania, Germany, England`
`Sofia, Bucharest, Berlin, London`
	`Bulgaria -> Sofia`
	`Romania -> Bucharest`
	`Germany -> Berlin`
	`England -> London`

5. Heroes Inventory

Using **comprehension** write a program that receives some **hero names**, **items** that need to be added in their inventory (item **name** and item **cost**) and then **prints** for each hero the total **amount of items** and the **total cost** of them.

Input

- On the first line you will receive the **names of the heroes** separated by comma and space **", "**
- On the next lines until the command **"End"**, you will be given **items** with their **cost** in the format **"{name}-{item}-{cost}"**. If an item **repeats** in a hero inventory, **ignore** it

Output

- For each hero print his **name**, the total **items** and the total **cost** of the items in the format: **"{name} -> Items: {items_count}, Cost: {items_cost}"**

Examples

Input	Output

`Peter, George`
`Peter-Sword-20`
`Peter-Shield-10`
`George-Gem-100`
`Peter-Sword-15`
`George-Sword-20`
`End`
	`Peter -> Items: 2, Cost: 30`
	`George -> Items: 2, Cost: 120`

6. Bunker

Using **comprehension** write a program that finds all the **amount** of all **items** in a bunker and their **average quantity**. On the first line you will be given **all the categories** of items in the bunker, then you will be given a number (**n**). On the next **"n"** lines you will be given items in the following format: **"{category} - {item_name} - quantity:{item_quantity};quality:{item_quality}"**. Store that information, you will need it later. After you received all the inputs, **print** the **total amount** of items (**sum the quantities**) in the format: **"Count of items: {count}"**. After that print the **average quality** of all items in the format: **"Average quality: {quality - formatted to the second digit}"**. Finally, **print** all of the **categories** with the **items** on **separate lines** in it in the format: **"{category} -> {item1}, {item2}…"**. For more clarification, see the example below.

Examples

Input	Output

`food, water, materials, metal`
`5`
`food - pizza - quantity:10;quality:5`
`water - mineral - quantity:5;quality:10`
`materials - wood - quantity:2;quality:5`
`metal - copper - quantity:3;quality:10`
`food - burgers - quantity:5;quality:2`
	`Count of items: 25`
	`Average quality: 8.00`
	`food -> pizza, burgers`
	`water -> mineral`
	`materials -> wood`
	`metal -> copper`