# Exam Preparation

## Calculator

Write a **function** that receives 3 parameters:

* **number**
* **operator** – text with the following options:**'+', '-', '/', '\*'**
* **another number**

Print the result of the calculation on the console formatted to the **second decimal** point.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5,  "+",  10 | 15.00 |
| 25.5,  "-",  3 | 22.50 |
| 9,  "/",  2 | 4.50 |
| 7,  "\*",  5 | 35.00 |

## Lowest Prices in Cities

You will be given several **towns**, with **products** and their **price**. You need to find **the lowest price** for **every product** and **the town it is sold at** for that price.

### Input

The **input** comes as an array of strings. Each element will hold data about a **town**, **product**, and **its price** at that town. The **town** and **product** will be **strings**, the **price** will be a **number**. The input will come in the following format:

{townName} | {productName} | {productPrice}

### Output

As **output**, you must print **each** **product** with its **lowest price** and **the town** at which the product is **sold at that** **price**. If **two towns share** the **same lowest price**, print the one that was **entered first**.   
The output, for every product, should be in the following format:

{productName} -> {productLowestPrice} ({townName})

The **order of output** in - **order of entrance**. See the examples for more info.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| ['Sample Town | Sample Product | 1000',  'Sample Town | Orange | 2',  'Sample Town | Peach | 1',  'Sofia | Orange | 3',  'Sofia | Peach | 2',  'New York | Sample Product | 1000.1',  'New York | Burger | 10'] | Sample Product -> 1000 (Sample Town)  Orange -> 2 (Sample Town)  Peach -> 1 (Sample Town)  Burger -> 10 (New York) |

1. **Clothing Magazine**

Your task is to create a repository that stores clothes by creating the classes described below.

### Cloth

Create class **Cloth** with the following properties:

* **color - string**
* **size - number**
* **type - string**

The class **constructor** should receive **color, size** and **type**.

Override the **toString()** method in the following format:  
**"Product: {type} with size {size}, color {color}"**

### Magazine

Create class **Magazine** that has **Clothes** (an array that stores the entity **Cloth**). All entities inside the repository have the **same properties**. The **Magazine** class should have the following **properties**:

* **type - string**
* **capacity – number**
* **clothes – Cloth[]**

The class **constructor** should receive **type** and **capacity**, also it should initialize the **clothes** with a new instance of the collection.Implement the following features:

* **Method addCloth(cloth: Cloth)** – **adds** an **entity** to the collection **if** **there** **is** **room** for it
* **Method removeCloth(color: string)** – removes a cloth by **given color,** if such **exists**, and **returns boolean** (**true** if it is removed, otherwise – **false**)
* **Method getSmallestCloth()** – **returns the Cloth with the smallest Size**
* **Method getCloth(color: string)** – **returns** the **Cloth** with the **given color**
* **Method getClothCount()** – **returns** the **number** of **clothes**
* **Method report()** – **returns** a **string** in the following **format** (print the clothes in **ordered by Size**):
  + **"{type} magazine contains:  
    {Cloth1}  
    {Cloth2}  
    (…)"**

**Constraints**

* The **color** and **size** of the clothes will be **always unique**.
* You will always have clothes added before receiving methods manipulating the Magazines’ clothes.

**Examples**

This is an example of how the **Magazine** class is **intended to be used**.

|  |
| --- |
| **Sample code usage** |
| **import { Magazine } from "./Magazine";**  **import { Cloth } from "./Cloth";**  **function main() {**  **// Initialize the repository (Magazine)**  **const magazine = new Magazine("Zara", 20);**  **// Initialize entity (Cloth)**  **const cloth1 = new Cloth("red", 36, "dress");**  **// Print Cloth**  **console.log(cloth1.toString());**  **// Product: dress with size 36, color red**  **// Add Cloth**  **magazine.addCloth(cloth1);**  **// Remove Cloth**  **console.log(magazine.removeCloth("black")); // false**  **const cloth2 = new Cloth("brown", 34, "t-shirt");**  **const cloth3 = new Cloth("blue", 32, "jeans");**  **// Add Cloth**  **magazine.addCloth(cloth2);**  **magazine.addCloth(cloth3);**  **// Get smallest cloth**  **const smallestCloth = magazine.getSmallestCloth();**  **console.log(smallestCloth?.toString());**  **// Product: jeans with size 32, color blue**  **// Get Cloth**  **const getCloth = magazine.getCloth("brown");**  **// Product: t-shirt with size 34, color brown**  **console.log(getCloth?.toString());**  **console.log(magazine.report());**  **// Zara magazine contains:**  **// Product: jeans with size 32, color blue**  **// Product: t-shirt with size 34, color brown**  **// Product: dress with size 36, color red**  **}**  **main();** |