# More 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.bg/Contests/1318

## 1. Class Laptop

Create a **class Laptop** that has the following properties:

**`info`** – object that contains:

- **`producer`** – string
- **`age`** – number
- **`brand`** – string

**`isOn`** – boolean (false by default)

**`turnOn`** – function that **sets the `isOn`** variable to **`true`**

**`turnOff`** – function that **sets the `isOn`** variable to **`false`**

**`showInfo`** – function that returns the **producer, age and brand as json**

**`quality`** – number (every time the laptop **is turned on/off the quality decreases by 1**)

**`getter price`** – number (**`800 - {age * 2} + (quality * 0.5)`**)

The **constructor** should receive the **info as an object and the quality**

### Examples

Test your class

| Input | Output |
|---|---|
| `let info = {producer: "Dell", age: 2, brand: "XPS"}`<br>`let laptop = new Laptop(info, 10)`<br>`laptop.turnOn()`<br>`console.log(laptop.showInfo())`<br>`laptop.turnOff()`<br>`console.log(laptop.quality)`<br>`laptop.turnOn()`<br>`console.log(laptop.isOn)`<br>`console.log(laptop.price)` | `{"producer":"Dell","age":2,"brand":"XPS"}`<br>`8`<br>`true`<br>`799.5` |

## 2. Flight Schedule

You will receive an **array** with **arrays.**

First array (**at index 0**) will hold all flights on **specific `sector`** in the airport. The second array (**at index 1**) will contain **new changed `statuses`** of **some** of the **`flights`** at this airport. The third array (**at index 2**) will have a single **string,** which will **be flight `status`** you need to check. When you put all flights into an **`object`,** and change the statuses depends on the new information on the second array. You must print all flights with the given status from the last **`array`**.

## Examples

| Input | Output |
|---|---|
| [['WN269 Delaware',<br>'FL2269 Oregon',<br>'WN498 Las Vegas',<br>'WN3145 Ohio',<br>'WN612 Alabama',<br>'WN4010 New York',<br>'WN1173 California',<br>'DL2120 Texas',<br>'KL5744 Illinois',<br>'WN678 Pennsylvania'],<br>['DL2120 Cancelled',<br>'WN612 Cancelled',<br>'WN1173 Cancelled',<br>'SK430 Cancelled'],<br>['Cancelled']<br>] | { Destination: 'Alabama', Status: 'Cancelled' }<br>{ Destination: 'California', Status: 'Cancelled' }<br>{ Destination: 'Texas', Status: 'Cancelled' } |
| [['WN269 Delaware',<br>'FL2269 Oregon',<br>'WN498 Las Vegas',<br>'WN3145 Ohio',<br>'WN612 Alabama',<br>'WN4010 New York',<br>'WN1173 California',<br>'DL2120 Texas',<br>'KL5744 Illinois',<br>'WN678 Pennsylvania'],<br>['DL2120 Cancelled',<br>'WN612 Cancelled',<br>'WN1173 Cancelled', | { Destination: 'Delaware', Status: 'Ready to fly' }<br>{ Destination: 'Oregon', Status: 'Ready to fly' }<br>{ Destination: 'Las', Status: 'Ready to fly' }<br>{ Destination: 'Ohio', Status: 'Ready to fly' }<br>{ Destination: 'New', Status: 'Ready to fly' }<br>{ Destination: 'Illinois', Status: 'Ready to fly' }<br>{ Destination: 'Pennsylvania', Status: 'Ready to fly' } |

| | |
|---|---|
| 'SK330 Cancelled'],<br>['Ready to fly']<br>] | |

## 3. School Register

In this problem you have to arrange all students by **grade**. You as the secretary of the school principal will process students and store them into a school register before the new school year hits. As a draft, you have a list of all the students from **last year** but mixed. Keep in mind that if a student has a lower grade than 3, he does not go into the next class. As result of your work, you have to print the entire school register **sorted** in **ascending order by grade** already filled with all the students from last year in format:

**{nextGrade} Grade**

**List of students: {All students in that grade}**

**Average annual grade from last year: {average annual grade on the entire class from last year}**

And empty row **{console.log}**

The input will be **array** with strings, each containing a student's name, last year's grade, and an annual grade. The **average annual grade from last year** should be **formatted to the second decimal point**.

### Examples

| Input | Output |
|---|---|
| ["Student name: Mark, Grade: 8, Graduated with an average score: 4.75",<br>"Student name: Ethan, Grade: 9, Graduated with an average score: 5.66",<br>"Student name: George, Grade: 8, Graduated with an average score: 2.83",<br>"Student name: Steven, Grade: 10, Graduated with an average score: 4.20",<br>"Student name: Joey, Grade: 9, Graduated with an average score: 4.90",<br>"Student name: Angus, Grade: 11, Graduated with an average score: 2.90",<br>"Student name: Bob, Grade: 11, Graduated with an average score: 5.15", | 9 Grade<br>List of students: Mark, Daryl<br>Average annual grade from last year: 5.35<br><br>10 Grade<br>List of students: Ethan, Joey, Bill<br>Average annual grade from last year: 5.52<br><br>11 Grade<br>List of students: Steven, Philip, Gavin<br>Average annual grade from last year: 4.42<br><br>12 Grade<br>List of students: Bob, Peter<br>Average annual grade from last year: 5.02 |

| | |
|---|---|
| "Student name: Daryl, Grade: 8, Graduated with an average score: 5.95",<br>"Student name: Bill, Grade: 9, Graduated with an average score: 6.00",<br>"Student name: Philip, Grade: 10, Graduated with an average score: 5.05",<br>"Student name: Peter, Grade: 11, Graduated with an average score: 4.88",<br>"Student name: Gavin, Grade: 10, Graduated with an average score: 4.00"] | |

## 4. Browser History

As an input you will receive **two parameters**: **an object and a string array.**

The object will be in format: **`{Browser Name}:{Name of the browser}, Open tabs:[…], Recently Closed: […], Browser Logs: […]`**. Your task is to fill in the object based on the actions we will get in the array of strings.

You can **open** any site in the world as many times as you like; if you do that add it to the open tabs.

You can **close** only these tabs you have **opened already**! If current action contains valid opened site, you should remove it from "**`Open Tabs`**" and put it into "**`Recently closed`**", otherwise **don't do anything!**

**Browser Logs** will hold every single **Valid** action which you did (Open and Close).

There"s a **special case** in which you can get an action that says: **`"Clear History and Cache".`** That means you should **empty the whole object**.

At the end print the object in format:

**`{Browser name}`**

**`Open Tabs: {[…]}`** // Joined by comma and space

**`Recently Closed: {[…]}`** // Joined by comma and space

**`Browser Logs: {[…]}`** // Joined by comma and space

### Examples

| Input | Output |
|---|---|
| {"Browser Name":"Google Chrome","Open Tabs":["Facebook","YouTube","Google Translate"],<br>"Recently Closed":["Yahoo","Gmail"],<br>"Browser Logs":["Open YouTube","Open Yahoo","Open Google | Google Chrome<br>Open Tabs: YouTube, Google Translate, StackOverFlow, Google<br>Recently Closed: Yahoo, Gmail, Facebook<br>Browser Logs: Open YouTube, Open Yahoo, Open Google Translate, Close Yahoo, Open Gmail, Close Gmail, Open Facebook, Close Facebook, Open StackOverFlow, Open Google |

| | |
|---|---|
| Translate","Close Yahoo","Open Gmail","Close Gmail","Open Facebook"]},<br>["Close Facebook", "Open StackOverFlow", "Open Google"] | |
| {"Browser Name":"Mozilla Firefox",<br>"Open Tabs":["YouTube"],<br>"Recently Closed":["Gmail", "Dropbox"],<br>"Browser Logs":["Open Gmail", "Close Gmail", "Open Dropbox", "Open YouTube", "Close Dropbox"]},<br>["Open Wikipedia", "Clear History and Cache", "Open Twitter"] | Mozilla Firefox<br>Open Tabs: Twitter<br>Recently Closed:<br>Browser Logs: Open Twitter |

## 5. Sequences

You are tasked with storing sequences of numbers. You will receive an **array of strings; each of them will contain** unknown amount of **arrays containing numbers,** from which you must store only the **unique** arrays (duplicate arrays should be discarded). An array is considered the **same** (**NOT unique**) if it contains the **same numbers** as another array**, regardless of their order**.

After storing all arrays, your program should print them back in **ascending** order based on their **length**, if two arrays have the same length they should be printed in **order of being received from the input**. Each individual array should be printed in **descending order** in the format **"[$a_1$, $a_2$, $a_3$,… $a_n$]"**. Check the examples bellow.

The **input** comes as an **array of strings** where **each entry is a JSON representing an array of numbers**.

The **output** should be printed on the console - each array printed on a new line in the format **"[a1, a2, a3,… an]"** , following the above mentioned ordering.

### Examples

| Input | Output |
|---|---|
| ["[-3, -2, -1, 0, 1, 2, 3, 4]",<br>"[10, 1, -17, 0, 2, 13]",<br>"[4, -3, 3, -2, 2, -1, 1, 0]"] | [13, 10, 2, 1, 0, -17]<br>[4, 3, 2, 1, 0, -1, -2, -3] |
| ["[7.14, 7.180, 7.339, 80.099]",<br>"[7.339, 80.0990, 7.140000, 7.18]",<br>"[7.339, 7.180, 7.14, 80.099]"] | [80.099, 7.339, 7.18, 7.14] |