

# More Exercise: Objects and Classes

Problems for exercise and homework for the ["JS Fundamentals" Course @ SoftUni](https://softuni.org/Courses/JS-Fundamentals).

Submit your solutions in the SoftUni judge system at: <https://judge.softuni.org/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** – a function that **sets the isOn** variable to **true**
- **turnOff** – a function that **sets the isOn** variable to **false**
- **showInfo** – a 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  |
|---|---|
| <pre>let info = {producer: "Dell", age: 2, brand: "XPS"} let laptop = new Laptop(info, 10) laptop.turnOn() console.log(laptop.showInfo()) laptop.turnOff() console.log(laptop.quality) laptop.turnOn() console.log(laptop.isOn) console.log(laptop.price)</pre> | <pre>{"producer":"Dell","age":2,"brand":"XPS"} 8 true 799.5</pre> |
| <pre>let info = {producer: "Lenovo", age: 1, brand: "Legion"} let laptop = new Laptop(info, 10) laptop.turnOn() console.log(laptop.showInfo()) laptop.turnOff() laptop.turnOn() laptop.turnOff()</pre>  | <pre>{"producer":"Lenovo","age":1,"brand":"Legion"} false</pre>   |

```
console.log(laptop.isOn)
```

## 2. Flight Schedule

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

The first array (**at index 0**) will hold all flights on a **specific sector** in the airport. The second array (**at index 1**) will contain **newly changed statuses** of **some** of the **flights** at this airport. The third array (**at index 2**) will have a single **string**, which will **be the 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**.

- If the value of the string obtained from the third array is **"Ready to fly"**:
  - then you must **print** flights that have **not changed** their **status** in the second array
  - and automatically **change** the status to **"Ready to fly"**
- Otherwise, print **only flights** that have **changed** their status.

## Examples

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

|   |  |
|---|--|
| <pre>'WN4010 New York', 'WN1173 California', 'DL2120 Texas', 'KL5744 Illinois', 'WN678 Pennsylvania'], ['DL2120 Cancelled', 'WN612 Cancelled', 'WN1173 Cancelled', 'SK330 Cancelled'], ['Ready to fly'] ]</pre> | <pre>{ Destination: 'Illinois', Status: 'Ready to fly' } { Destination: 'Pennsylvania', Status: 'Ready to fly' }</pre> |
|---|--|

### 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 score than 3, he does not go into the next class. As a 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 the format:

```
`{nextGrade} Grade
```

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

```
Average annual score from last year: {average annual score on the entire class from last year}`
```

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

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

### Examples

| Input   | Output   |
|---|--|
| <pre>[ "Student name: Mark, Grade: 8, Graduated with an average score: 4.75", "Student name: Ethan, Grade: 9, Graduated with an average score: 5.66", "Student name: George, Grade: 8, Graduated with an average score: 2.83", "Student name: Steven, Grade: 10, Graduated with an average score: 4.20", "Student name: Joey, Grade: 9, Graduated with an average score: 4.90", ]</pre> | <pre>9 Grade List of students: Mark, Daryl Average annual score from last year: 5.35  10 Grade List of students: Ethan, Joey, Bill</pre> |

|   |   |
|---|---|
| "Student name: Angus, Grade: 11,<br>Graduated with an average score: 2.90",<br>"Student name: Bob, Grade: 11,<br>Graduated with an average score: 5.15",<br>"Student name: Daryl, Grade: 8,<br>Graduated with an average score: 5.95",<br>"Student name: Bill, Grade: 9,<br>Graduated with an average score: 6.00",<br>"Student name: Philip, Grade: 10,<br>Graduated with an average score: 5.05",<br>"Student name: Peter, Grade: 11,<br>Graduated with an average score: 4.88",<br>"Student name: Gavin, Grade: 10,<br>Graduated with an average score: 4.00"<br>]   | Average annual score from last<br>year: 5.52<br><br>11 Grade<br>List of students: Steven, Philip,<br>Gavin<br>Average annual score from last<br>year: 4.42<br><br>12 Grade<br>List of students: Bob, Peter<br>Average annual score from last<br>year: 5.02  |
| [<br>'Student name: George, Grade: 5,<br>Graduated with an average score: 2.75',<br>'Student name: Alex, Grade: 9,<br>Graduated with an average score: 3.66',<br>'Student name: Peter, Grade: 8,<br>Graduated with an average score: 2.83',<br>'Student name: Bobby, Grade: 5,<br>Graduated with an average score: 4.20',<br>'Student name: John, Grade: 9,<br>Graduated with an average score: 2.90',<br>'Student name: Steven, Grade: 2,<br>Graduated with an average score: 4.90',<br>'Student name: Darsy, Grade: 1,<br>Graduated with an average score: 5.15'<br>] | 2 Grade<br>List of students: Darsy<br>Average annual score from last<br>year: 5.15<br><br>3 Grade<br>List of students: Steven<br>Average annual score from last<br>year: 4.90<br><br>6 Grade<br>List of students: Bobby<br>Average annual score from last<br>year: 4.20<br><br>10 Grade<br>List of students: Alex<br>Average annual score from last<br>year: 3.66 |

## 4. Browser History

As 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 the current action contains a 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 is a **special case** in which you can get an action that says: **"Clear History and Cache"**. That means you should **empty the whole object**.

In the end, print the object in the 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   |
|--|--|
| <pre>{ "Browser Name": "Google Chrome", "Open Tabs": ["Facebook", "YouTube", "Google Translate"], "Recently Closed": ["Yahoo", "Gmail"], "Browser Logs": ["Open YouTube", "Open Yahoo", "Open Google Translate", "Close Yahoo", "Open Gmail", "Close Gmail", "Open Facebook"] }, ["Close Facebook", "Open StackOverFlow", "Open Google"]</pre> | <pre>Google Chrome Open Tabs: YouTube, Google Translate, StackOverFlow, Google Recently Closed: Yahoo, Gmail, Facebook Browser Logs: Open YouTube, Open Yahoo, Open Google Translate, Close Yahoo, Open Gmail, Close Gmail, Open Facebook, Close Facebook, Open StackOverFlow, Open Google</pre> |
| <pre>{ "Browser Name": "Mozilla Firefox", "Open Tabs": ["YouTube"], "Recently Closed": ["Gmail", "Dropbox"], "Browser Logs": ["Open Gmail", "Close Gmail", "Open Dropbox", "Open YouTube", "Close Dropbox"] }, ["Open Wikipedia", "Clear History and Cache", "Open Twitter"]</pre>   | <pre>Mozilla Firefox Open Tabs: Twitter Recently Closed: Browser Logs: Open Twitter</pre>  |

## 5. Sequences

You are tasked with storing sequences of numbers. You will receive an **array of strings**; each of them will contain an 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 array should be printed in **descending order** in the format **"[a<sub>1</sub>, a<sub>2</sub>, a<sub>3</sub>,... a<sub>n</sub>]"**. Check the examples below.

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  |
|---|---|
| <code>"[-3, -2, -1, 0, 1, 2, 3, 4]",<br/>"[10, 1, -17, 0, 2, 13]",<br/>"[4, -3, 3, -2, 2, -1, 1, 0]"</code>             | <code>[13, 10, 2, 1, 0, -17]<br/>[4, 3, 2, 1, 0, -1, -2, -3]</code> |
| <code>"[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]"</code> | <code>[80.099, 7.339, 7.18, 7.14]</code>                            |