# More Exercises: Objects, Associative Arrays, Maps, Sets

Problems for exercises and homework for the [“JavaScript Fundamentals” course @ SoftUni](https://softuni.bg/trainings/2080/js-fundamentals-september-2018). Submit your solutions in the SoftUni judge system at <https://judge.softuni.bg/Contests/1277/Objects-and-JSONS-More-Exercises>.

**This exercise is not included in calculating the homework bonus!**

## Followers

You need to write a function that allows **users in a website to follow one another**. In order for users to be able to follow one another, **they have to be registered in the website**. There are four more possible input strings:  
 - "Welcome, {user}" – if the user is **not registered**, you need to **register him** - "{user} followed {another user}" – the **first user** starts **following the other one**, only if **both of them exist.**

When you finish reading the input, print the results as follows:  
 "Total users registered: {total users}"  
 "1. {first user}: {following} following, {followers} followers"  
 "\* {follower one}"  
 "\* {follower two}"  
 "2. {second user}: {following} following, {followers} followers"  
 "3. {second user}: {following} following, {followers} followers"

Users must be sorted by **count of followers in descending**, then **by username**. For the **user with most followers**, print **their names ordered alphabetically**.

### Input

The **input** will come as an **array of strings.**

### Output

Print the result as described above.

### Constraints

* There will be no invalid input.
* There will be no situation where two users have equal amount of followers and equal amount of followings
* The subscribers will be strings.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Welcome, EmilConrad  Welcome, VenomTheDoctor  Welcome, Saffrona  Saffrona followed EmilConrad  Saffrona followed VenomTheDoctor  EmilConradfollowed VenomTheDoctor  VenomTheDoctorfollowed VenomTheDoctor  Saffrona followed EmilConrad | Total users registered: 3  1. VenomTheDoctor : 0 following, 2 followers  \* EmilConrad  \* Saffrona  2. EmilConrad : 1 following, 1 followers  3. Saffrona : 2 following, 0 followers |
| Welcome, JennaMarbles  JennaMarbles followed Zoella  Welcome, AmazingPhil  JennaMarbles followed AmazingPhil  Welcome, Zoella  Welcome, JennaMarbles  Zoella followed AmazingPhil  Christy followed Zoella  Zoella followed Christy  Welcome, JacksGap  JacksGap followed JennaMarbles  Welcome, PewDiePie  Welcome, Zoella | Total users registered: 5  1. AmazingPhil : 0 following, 2 followers  \* JennaMarbles  \* Zoella  2. Zoella : 1 following, 1 followers  3. JennaMarbles : 2 following, 1 followers  4. PewDiePie : 0 following, 0 followers  5. JacksGap : 1 following, 0 followers |

## Travellers Log

Write a function that stores information about different people travelling in different countries and visiting different landmarks. They will also be spending money to do that, so you need to keep track of their money too! There are two types of input:  
 **"{name} gets {money}"** – if the traveller does not exist already, add him and the money to his account, otherwise just add the money   
 **"{name} visited the {landmark} in {country} – costs {money}"** – if the traveller does **not exist** yet, **add him**, if the traveller has yet **not been in that country**, **add it to his log with the landmark**; if the person **has been to the country**, but **not yet to this landmark**, **add the landmark** to the country in his log, **otherwise ignore**. Keep in mind that all of this happens, **only if the traveller has enough money**! If the traveller **does not have enough money**, print the following message: **"Not enough money to visit {landmark}".** If you add anything to his log, **reduce the money he has with by the given amount**.

### Input

The input will come as an **array of strings**

### Output

After processing all of the data, print the result in the following format:  
**"{name} visited {countries count} countries and has {money left} money left"  
"- {country one} -> {total landmarks} landmarks"  
"-- {landmark one}"**  
**Travellers** should be **sorted** by **total countries visited** by **descending**. **Countries** should be sorted by **total landmarks in descending** and for **each country the landmarks should be sorted alphabetically**.

|  |  |
| --- | --- |
| **Input** | **Output** |
| ['Peter gets 100',  'Peter visited the StatueOfLiberty in USA - 50',  'Bill gets 250',  'Tim visited the ChristTheRedeemer in Brazil - 150',  'Bill gets 400',  'Bill visited the MountFuji in Japan - 600',  'Bill visited the TeatroAmazonas in Brazil - 50',  'Bill gets 150',  'Bill visited the ChristTheRedeemer in Brazil - 150',  'Tim gets 500',  'Bill visited the StatueOfLiberty in USA - 440',  'Tim visited the StatueOfLiberty in USA - 440',  'Maria gets 650',  'Maria visited the StatueOfLiberty in USA - 440',  'Maria visited the CapeCod in USA - 100'] | Not enough money to visit ChristTheRedeemer  Not enough money to visit StatueOfLiberty  Bill visited 2 countries and has 0 money left  - Brazil -> 2 landmarks  -- ChristTheRedeemer  -- TeatroAmazonas  - Japan -> 1 landmarks  -- MountFuji  Peter visited 1 countries and has 50 money left  - USA -> 1 landmarks  -- StatueOfLiberty  Tim visited 1 countries and has 60 money left  - USA -> 1 landmarks  -- StatueOfLiberty  Maria visited 1 countries and has 110 money left  - USA -> 2 landmarks  -- CapeCod  -- StatueOfLiberty |
| ['Peter gets 100',  'Peter visited the StatueOfLiberty in USA - 50',  'Bill gets 250',  'Bill gets 400',  'Peter gets 150',  'Peter visited the ChristTheRedeemer in Brazil - 150'] | Peter visited 2 countries and has 50 money left  - USA -> 1 landmarks  -- StatueOfLiberty  - Brazil -> 1 landmarks  -- ChristTheRedeemer  Bill visited 0 countries and has 650 money left |

## School Grades

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 to hit. Аs a draft you have a list of all the students from **last year** but mixed. Кeep in mind that if a student has a lesser 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 student: {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}"**

### Input

The input will be **array**, **with strings**, each containing a student's name, last year's grade, and an annual grade.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| ['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',  'Student name: Angus, Grade: 11, Graduated with an average score: 2.90',  'Student name: Bob, Grade: 11, Graduated with an average score: 5.15',  'Student name: Daryl, Grade: 8, Graduated with an average score: 5.95',  'Student name: Bill, Grade: 9, Graduated with an average score: 6.00',  'Student name: Philip, Grade: 10, Graduated with an average score: 5.05',  'Student name: Peter, Grade: 11, Graduated with an average score: 4.88',  'Student name: Gavin, Grade: 10, Graduated with an average score: 4.00'] | 9 Grade  List of students: Mark, Daryl  Average annual grade from last year: 5.35  10 Grade  List of students: Ethan, Joey, Bill  Average annual grade from last year: 5.52  11 Grade  List of students: Steven, Philip, Gavin  Average annual grade from last year: 4.42  12 Grade  List of students: Bob, Peter  Average annual grade from last year: 5.02 |

## Browser Logger

As input you will receive an array with **2 parameters**, **OBJECT and 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 string 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 have to **clear your 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 space**

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

**Browser Logs: {[…]} // Joined by space"**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| {"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'] | 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 |
| {"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'] | Mozilla Firefox  Open Tabs: Twitter  Recently Closed:  Browser Logs: Open Twitter |

## Catalogue

You have to create a sorted catalogue of store products. You will be given the products’ names and prices. You need to order them by **alphabetical order**.

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**.

As **output** you must print all the products in a specified format. They must be ordered **exactly as specified above**. The products must be **divided into groups**, by the **initial of their name**. The **group’s initial should be printed**, and after that the products should be printed with **2 spaces before their names**. For more info check the examples.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| ["Appricot : 20.4",  "Fridge : 1500",  "TV : 1499",  "Deodorant : 10",  "Boiler : 300",  "Apple : 1.25",  "Anti-Bug Spray : 15",  "T-Shirt : 10"] | A  Anti-Bug Spray: 15  Apple: 1.25  Appricot: 20.4  B  Boiler: 300  D  Deodorant: 10  F  Fridge: 1500  T  T-Shirt: 10  TV: 1499 |

## 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,** who will **be flight status** you need to check. When you put all flights into a **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',  'FL2269 Oregon',  'WN498 Las Vegas',  'WN3145 Ohio',  'WN612 Alabama',  'WN4010 New York',  'WN1173 California',  'DL2120 Texas',  'KL5744 Illinois',  'WN678 Pennsylvania'],  ['DL2120 Cancelled',  'WN612 Cancelled',  'WN1173 Cancelled',  'SK430 Cancelled'],  ['Cancelled']  ] | { Destination: 'Alabama', Status: 'Cancelled' }  { Destination: 'California', Status: 'Cancelled' }  { Destination: 'Texas', Status: 'Cancelled' } |
| [['WN269 Delaware',  'FL2269 Oregon',  'WN498 Las Vegas',  'WN3145 Ohio',  'WN612 Alabama',  'WN4010 New York',  'WN1173 California',  'DL2120 Texas',  'KL5744 Illinois',  'WN678 Pennsylvania'],  ['DL2120 Cancelled',  'WN612 Cancelled',  'WN1173 Cancelled',  'SK330 Cancelled'],  ['Ready to fly']  ] | { Destination: 'Delaware', Status: 'Ready to fly' }  { Destination: 'Oregon', Status: 'Ready to fly' }  { Destination: 'Las', Status: 'Ready to fly' }  { Destination: 'Ohio', Status: 'Ready to fly' }  { Destination: 'New', Status: 'Ready to fly' }  { Destination: 'Illinois', Status: 'Ready to fly' }  { Destination: 'Pennsylvania', Status: 'Ready to fly' } |