## Problem 4 – Minka the JavaScript Goddess

Minka **really** loves JavaScript. She has already solved all the ordinary JavaScript exams, so she decided to move on. Her next mission is to deal with tasks, designed to be solved in C#. These tasks are in format

**Name & Type & Task Number & Score & Lines of code**

Your job is to group all the available data in a correct format.

For each **Task Number,** create an object.

Each of these objects should hold data for all the tasks that correspond to that number.

Display that data in an array that holds the **Name** and **Type.**

The next-to-last element of the object should be a floating-point number that represents the average score for these tasks.

The last part of the object is an integer number, which represents the total number of code lines.

The last part of your task is to **sort** the data by **average score**, in descending order. If the average score of two tasks is the same, **sort** by total **number of lines**, in ascending order.

Each of the arrays should be sorted alphabetically by **name**.

Two or more tasks with the same name and type are considered **different**.

### Input

The input is passed to the first JavaScript function found in your code as **array of strings** holding the table lines. The input data will always be valid and in the format described. There is no need to check it explicitly.

### Output

Print on the console a **JSON string** that holds the **task numbers**; for each number: an array for each individual task that holds **name** and **type.** Add **average** score and total **lines** of code for each inner object. Please follow exactly the **JSON format** from the example below. The average score should be rounded to two digits after the decimal point. Trailing zeroes should be **removed**. For example, **95**.**11** is displayed as **95**.**11**, **95**.**10** is displayed as **95**.**1**, and **95**.**00** is displayed as **95**.

### Constraints

* The numbers of **tasks** is between 1 and 100.
* Allowed working time: 0.25 seconds. Allowed memory: 16 MB.

### Examples

|  |
| --- |
| **Input** |
| Array Matcher & strings & 4 & 100 & 38  Magic Wand & draw & 3 & 100 & 15  Dream Item & loops & 2 & 88 & 80  Knight Path & bits & 5 & 100 & 65  Basket Battle & conditionals & 2 & 100 & 120  Torrent Pirate & calculations & 1 & 100 & 20  Encrypted Matrix & nested loops & 4 & 90 & 52  Game of bits & bits & 5 & 100 & 18  Fit box in box & conditionals & 1 & 100 & 95  Disk & draw & 3 & 90 & 15  Poker Straight & nested loops & 4 & 40 & 57  Friend Bits & bits & 5 & 100 & 81 |
| **Output** |
| {"Task 1":{"tasks":[{"name":"Fit box in box","type":"conditionals"},{"name":"Torrent Pirate","type":"calculations"}],"average":100,"lines":115},"Task 5":{"tasks":[{"name":"Friend Bits","type":"bits"},{"name":"Game of bits","type":"bits"},{"name":"Knight Path","type":"bits"}],"average":100,"lines":164},"Task 3":{"tasks":[{"name":"Disk","type":"draw"},{"name":"Magic Wand","type":"draw"}],"average":95,"lines":30},"Task 2":{"tasks":[{"name":"Basket Battle","type":"conditionals"},{"name":"Dream Item","type":"loops"}],"average":94,"lines":200},"Task 4":{"tasks":[{"name":"Array Matcher","type":"strings"},{"name":"Encrypted Matrix","type":"nested loops"},{"name":"Poker Straight","type":"nested loops"}],"average":76.67,"lines":147}} |