More Exercise: Arrays and Matrices

Problems for exercise and homework for the "JavaScript Fundamentals Course@SoftUni". Submit your solutions in the SoftUni judge system at https://judge.softuni.bg/Contests/1471.

1. Car Category

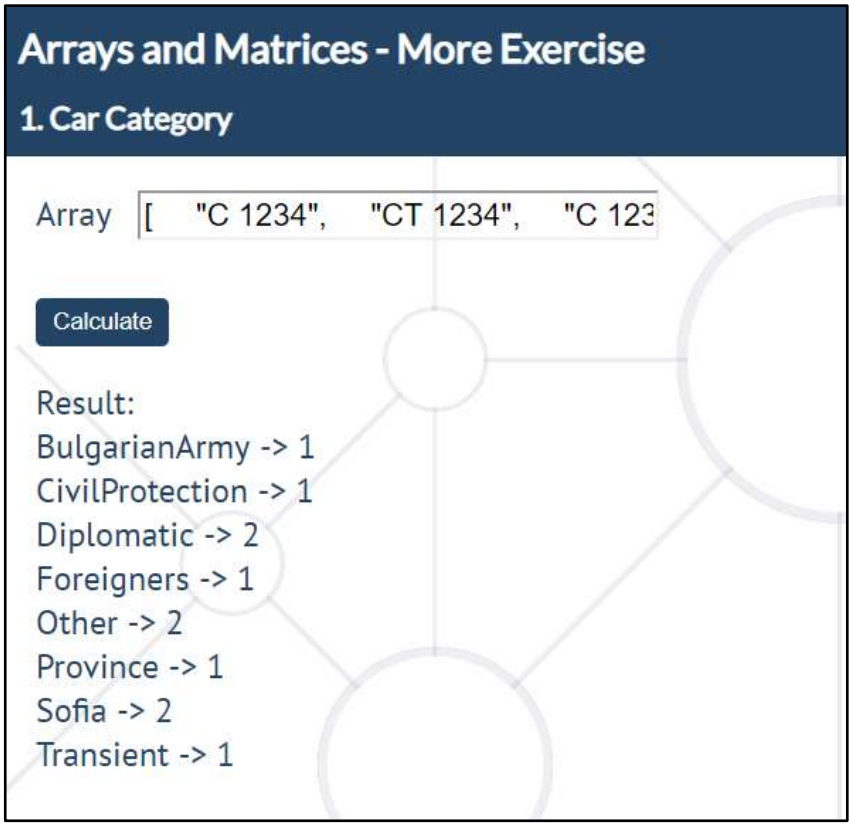
Write a function that takes as parameter an array of strings, which represent car numbers. The possible categories of car numbers are:

- **BulgarianArmy** - **"BA 123 456"** -> starts with "BA" and is followed by two 3-digit numbers.
- **CivilProtection** - **"CP 12 345"** -> starts with "CP" and is followed by two number groups, the first one with 2-digits and the second one with 3-digits.
- **Diplomatic** - **"C 1234", "CT 1234"** -> starts with "C" or "CT" and is followed by a 4-digit number.
- **Foreigners** - **"XX 1234"** -> starts with "XX" and is followed by a 4-digit number.
- **Transient** - **"123 B 123"** -> starts with 3 digits than a single letter and another 3 digits.
- **Sofia** - **"CA 1234 CA", "CB 1234 CB", "C 1234 C", "C 1234 CA"** -> starts with "C", "CA" or "CB" and is followed by 4-digits, and ends with one or two letters.
- **Province** – **"K 2412 DX"**-> starts with one or two letters, then 4 digits and one or two letters again.
- **Other** – all that do not meet the conditions above.

Display the cars **sorted** by count of numbers in **descending** then by category name. **Each row is a paragraph text.**

Examples

Input	Output
["C 1234", "CT 1234", "C 1234 C", "CP 12 345", "CA 1234 CA", "BA 123 456", "XX 1234", "123 B 123", "B 1234 BB", "11111111", "C1111111"]	BulgarianArmy -> 1 CivilProtection -> 1 Diplomatic -> 2 Foreigners -> 1 Other -> 2 Province -> 1 Sofia -> 2 Transient -> 1



2. Book Shelfs

Write a function that stores information about **shelves** and the **books in the shelves**. Each shelf has an **Id** and a **genre** of books that can be in it. Each book has a **title**, an **author** and **genre**. The input comes as an **array of strings**. They will be in the format:

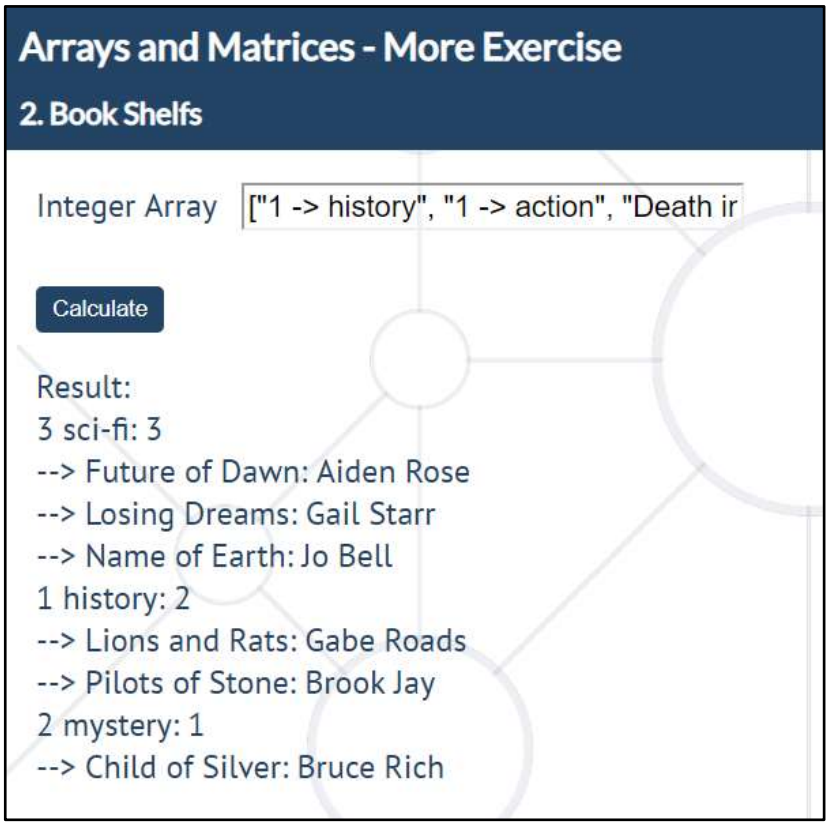
- **`"{shelf id} -> {shelf genre}"`** – create a shelf **if the id is not taken**
- **`"{book title}: {book author}, {book genre}"`** – if a shelf with that **genre exists**, add the book to the shelf

After finished reading input, sort the shelves by **count of books** in it in **descending**. For each shelf sort the **books by title** in ascending. Then display them in the following format each in a separate **<p>**.

- ```
  "{shelveOne id} {shelf genre}: {books count}
  --> {bookOne title}: {bookOne author}
  --> {bookTwo title}: {bookTwo author}
  …
  {shelveTwo id} {shelf genre}: {books count}
  …"
  ```
  ```

Examples

Input	Output
["1 -> history", "1 -> action", "Death in Time: Criss Bell, mystery", "2 -> mystery", "3 -> sci-fi", "Child of Silver: Bruce Rich, mystery", "Hurting Secrets: Dustin Bolt, action", "Future of Dawn: Aiden Rose, sci-fi", "Lions and Rats: Gabe Roads, history", "2 -> romance", "Effect of the Void: Shay B, romance", "Losing Dreams: Gail Starr, sci-fi", "Name of Earth: Jo Bell, sci-fi", "Pilots of Stone: Brook Jay, history"]	3 sci-fi: 3 --> Future of Dawn: Aiden Rose --> Losing Dreams: Gail Starr --> Name of Earth: Jo Bell 1 history: 2 --> Lions and Rats: Gabe Roads --> Pilots of Stone: Brook Jay 2 mystery: 1 --> Child of Silver: Bruce Rich



3. Card Wars

The game is played by the rules of the game "War".

There are **two players**, each with **13** cards. *(The cards are valid, as the standard 52-card deck.)*

The input is an **array of 2 arrays**. First starts the first player in order of input with the card at 0 index. Then it's second one's turn. The process is repeated until the end of the game.

The player with the **higher** card takes both of the cards, by adding them at the **end** of his array (first the card of the first player, then the card of the second).

The winner is either the one who collects all cards, or the one who has collected more cards than the other on 20th turn.

2, 3, 4, 5, 6, 7, 8, 9, 10, J (11), Q (12), K (13), A (14)

Then it's time to play.

There will be **no case**, where the two players will have the same cards.

Input

Array of arrays

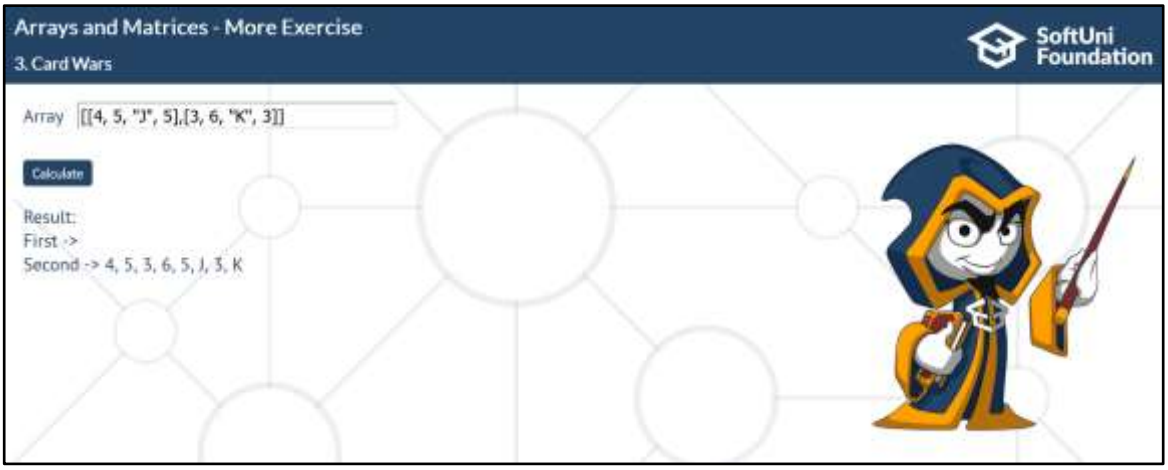
Output

The output consists of the cards of both players, as shown below:

```
<p>First -> 8, 7, 7, 7, 3, 3, K, 10, Q, J, 9, A, 4, Q, K, Q, 4, 2</p>
<p>Second -> 10, 6, J, 2, 3, 9, A, 5</p>
```

Examples

Input	Output
`[[4, 5, "J", 5],[3, 6, "K", 3]]`	`First ->` `Second -> 4, 5, 3, 6, 5, J, 3, K`
`[[10, "K", "Q", 2, 4], ["A", 3, "J", 2, 8]]`	`First -> 3, J, 2, K, 2, 10, 4, Q, 8` `Second -> A`



4. Radio Crystals

It's time to put your skills to work for the war effort – creating management software for a radio transmitter factory. Radios require a finely tuned quartz crystal in order to operate at the correct frequency. The resource used to produce them is quartz ore that comes in big chunks and needs to undergo several processes, before it reaches the desired properties.

You need to write a JS program that monitors the current thickness of the crystal and recommends the next procedure that will bring it closer to the desired frequency. To reduce waste and the time it takes to make each crystal your program needs to complete the process with the least number of operations. Each operation takes the same amount of time, but since they are done at different parts of the factory, the crystals have to be transported and thoroughly washed every time an operation different from the previous must be performed, so this must also be taken into account. When determining the order, always attempt to start from the operation that removes the largest amount of material.

The different operations you can perform are the following:

- **Cut** – cuts the crystal in 4
- **Lap** – removes 20% of the crystal's thickness
- **Grind** – removes 20 microns of thickness
- **Etch** – removes 2 microns of thickness
- **X-ray** – increases the thickness of the crystal by 1 micron; this operation can only be done once!
- **Transporting and washing** – removes any imperfections smaller than 1 micron (round down the number); do this after every batch of operations that remove material

At the beginning of your program, you will receive a **number** representing the **desired final thickness** and a **series of numbers**, representing **the thickness of crystal ore** in **microns**. Process each chunk and print the order of operations and number of times they need to be repeated to bring them to the desired thickness.

The **input** comes as a numeric array with a variable number of elements. The first number is the target thickness and all following numbers are the thickness of different chunks of quartz ore.

The **output** is the order of operation and how many times they are repeated, every operation in a **separate paragraph**. See the examples for more information.

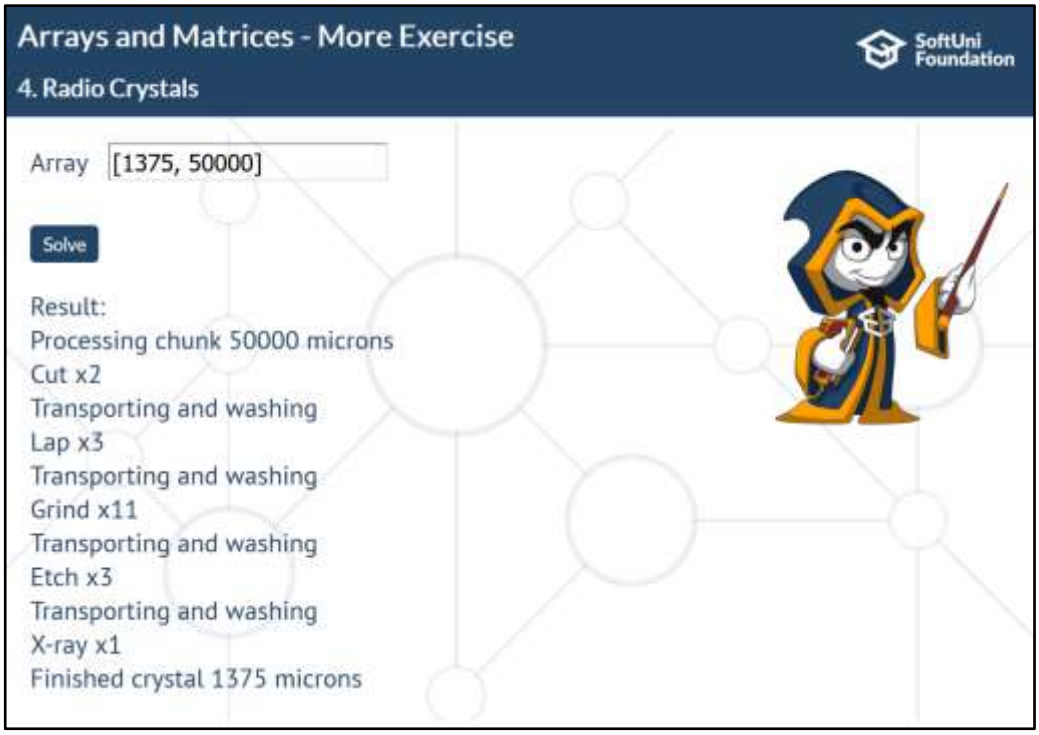
Examples

Input	Output
[1375, 50000]	Processing chunk 50000 microns Cut x2 Transporting and washing Lap x3 Transporting and washing Grind x11 Transporting and washing Etch x3 Transporting and washing X-ray x1 Finished crystal 1375 microns

Explanation

The operation that will remove the most material is always cutting – it removes three quarters of the chunk. Starting from 50000, if we perform it twice, we bring the chunk down to 3125. If we cut again, the chunk will be 781.25, which is less than the desired thickness, so we move to the next operation, but we first round down the number (transporting & washing). Next, we lap the chunk – after three operations, the crystal reaches 1600 microns. One more lapping would take it to 1280, so we move on to the next operation instead. We do the same check with grinding, and finally by etching 2 times, the crystal has reached 1376 microns, which is one more than desired. We don't have an operation which only takes away 1 micron, so instead we etch once more to get to 1374, wash and then x-ray to add 1 micron, which brings us to the desired thickness.

Input	Output
[1000, 4000, 8100]	Processing chunk 4000 microns Cut x1 Transporting and washing Finished crystal 1000 microns Processing chunk 8100 microns Cut x1 Transporting and washing Lap x3 Transporting and washing Grind x1 Transporting and washing Etch x8 Transporting and washing Finished crystal 1000 microns



5. *DNA Helix

Write a JS program that prints a DNA helix with length, specified by the user. The helix has a repeating structure, but the symbol in the chain follows the sequence *ATCGTTAGGG*. See the examples for more information.

The **input** comes as a single number. It represents the length of the required helix.

The **output** is the completed structure, displayed in the DOM, each line in separate **paragraph**.

Examples

Input	Output
4	**AT** *C--G* T----T *A--G*

Input	Output
10	**AT** *C--G* T----T *A--G* **GG** *A--T* C----G *T--T* **AG** *G--G*