# Tagram

You will receive **several input lines** in one of the following formats:

* "{username} -> {tag} -> {likes}"
* "ban {username}"

The username and tag are strings. **Likes** will be an integer number. You need to keep track of **every user**.

When you receive a **user**, a **tag** and **likes**, register the user if **he isn't present**, **otherwise add** the tag and the likes. If the user has already used the tag just add the likes to it.

If you receive **"ban {username}"** and **the username exists**, remove him from the database.

You should end your program when you receive the command "end". At that point you should print the users, **ordered by total likes in desecending order, then ordered by the tags’ count in ascending order**. **Foreach** player print their tag and likes.

## Input / Constraints

* The input comes in the form of commands in one of the formats specified above.
* Username and tag **will always be one word string, containing no whitespaces**.
* Likes will be an **integer** in the **range [0, 1000]**.
* There will be **no invalid** input lines.
* The programm ends when you receive the command "end".

## Output

* The output format for each player is:

"{username}"

"- {tag}: {likes}"

## Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Katty -> healthy -> 50  Elvin -> food -> 20  John -> music -> 30  Katty -> fitness -> 100  end | Katty  - healthy: 50  - fitness: 100  John  - music: 30  Elvin  - food: 20 |
| **Input** | **Output** |
| Monica -> music -> 100  Monica -> dance -> 50  John -> chill -> 200  Santa -> angry -> 300  ban Santa  Joshua -> football -> 500  end | Joshua  - football: 500  John  - chill: 200  Monica  - music: 100  - dance: 50 |
| Ani -> A1 -> 100  Bobi -> B2 -> 100  Bobi -> BB2 -> 150  Ani -> AA1 -> 100  Ani -> AAA1 -> 50  end | Bobi  - B2: 100  - BB2: 150  Ani  - A1: 100  - AA1: 100  - AAA1: 50 |