Exercise: ц

Tasks for exercise in class and for homework to the course ["Programming Advanced for QA" @ SoftUni](#)
Test your tasks in the Judge system: https://judge.softuni.org/Contests/4487

1. Students

Create a program that sorts some students by their grade in descending order. Each student should have:

- **First name** (string)
- **Last name** (string)
- **Grade** (a floating-point number)

Input

- On the first line, you will receive a number **n** - the **count of all students.**
- On the next **n** lines, you will be receiving information about these students in the following format: **`"{first name} {second name} {grade}"`.**

Output

- Print out the information about each student in the following format: **`"{first name} {second name}: {grade}"`.**

Example

Input	Output
4 Lakia Eason 3.90 Prince Messing 5.49 Akiko Segers 4.85 Rocco Erben 6.00	Rocco Erben: 6.00 Prince Messing: 5.49 Akiko Segers: 4.85 Lakia Eason: 3.90
3 Mary Elizabeth 4.22 Li Xiao 5.74 Liz Smith 4.87	Li Xiao: 5.74 Liz Smith: 4.87 Mary Elizabeth: 4.22

2. Articles

Create a **class `Article`** with the following properties:

- **`Title`** – a string
- **`Content`** – a string
- **`Author`** – a string

The class should have a constructor and the following methods:

- **`Edit (new content)`** – change the old content with the new one
- **`ChangeAuthor (new author)`** – change the author
- **`Rename (new title)`** – change the title of the article
- Override the **`ToString`** method – print the article in the following format: **`"{title} - {content}: {author}"`**

Create a program that reads an article in the following format **"{title}, {content}, {author}"**. On the next line, you will receive a number **n,** representing the number of commands, which will follow after it. On the next **n lines,** you will be receiving the following commands:

- **"Edit: {new content}"**
- **"ChangeAuthor: {new author}"**
- **"Rename: {new title}"**

In the end, print the final state of the article.

Example

Input	Output
some title, some content, some author 3 Edit: better content ChangeAuthor: better author Rename: better title	better title - better content: better author
Fight club, love story, Martin Scorsese 2 Edit: underground fight club that evolves into much more ChangeAuthor: Chuck Palahniuk	Fight club - underground fight club that evolves into much more: Chuck Palahniuk

3. Teamwork Projects

It's time for the teamwork projects and you are responsible for gathering the teams. First, you will receive an integer – the **count** of the **teams** you will have to **register**. You will be given a **user** and a **team**, separated with **"-"**. The user is the **creator** of **the team**. For every newly created team you should **print** a message:

"Team {teamName} has been created by {user}!".

Next, you will receive a user with a team, separated with **"->"**, which means that the user wants to **join** that **team**. Upon receiving the command: **"end of assignment"**, you should print **every team**, **ordered** by the **count** of its **members** (**descending**) and then by **name** (**ascending**). For each team, you have to print its members **sorted** by name (**ascending**). However, there are several **rules**:

- If a user tries to **create** a team more than once, a message should be displayed:
  - **"Team {teamName} was already created!"**
- A creator of a team **cannot create** another team – the following message should be thrown:
  - **"{user} cannot create another team!"**
- If a user tries to **join** a non-existent team, a message should be displayed:
  - **"Team {teamName} does not exist!"**
- A member of a team **cannot join** another team – the following message should be thrown:
  - **"Member {user} cannot join team {team Name}!"**
- In the end, teams with **zero** members (with **only a creator**) should **disband** and you have to print them **ordered by name in ascending order**.
- Every **valid** team should be printed ordered by **name** (ascending) in the following format:

```
"{teamName}
```

```
- {creator}
-- {member}…"
```

Examples

Input	Output	Comments

`2`
`John-PowerPuffsCoders`
`Tony-Tony is the best`
`Peter->PowerPuffsCoders`
`Tony->Tony is the best`
`end of assignment`
	`Team PowerPuffsCoders has been created by John!`
	`Team Tony is the best has been created by Tony!`
	Member Tony cannot join team Tony is the best!
	`PowerPuffsCoders`
	`- John`
	`-- Peter`
	`Teams to disband:`
	`Tony is the best`
		Tony created a team, which he attempted to join later and this action resulted in throwing a certain message. Since nobody else tried to join his team, the team had to **disband**.

`3`
`Tanya-CloneClub`
`Helena-CloneClub`
`Tedy-SoftUni`
`George->softUni`
`George->SoftUni`
`Tatyana->Leda`
`John->SoftUni`
`Cossima->CloneClub`
`end of assignment`
	`Team CloneClub has been created by Tanya!`
	`Team CloneClub was already created!`
	`Team SoftUni has been created by Tedy!`
	`Team softUni does not exist!`
	`Team Leda does not exist!`
	`SoftUni`
	`- Tedy`
	`-- George`
	`-- John`
	`CloneClub`
	`- Tanya`
	`-- Cossima`
	`Teams to disband:`
		Note that when a user joins a team, you should first check if the team exists and then check if the user is already in a team:
		
		Tanya has created CloneClub, then she tried to join a non-existent team and the concrete message was displayed.

4. Pokemon Trainer

Define a class **`Trainer`** and a class **`Pokemon`**.

`Trainers` have:

- **`Name`**
- **`Number of badges`**
- **`A collection of pokemon`**

`Pokemon` have:

- **`Name`**
- **`Element`**
- **`Health`**

All values are **mandatory**. Every `Trainer` **starts with 0 badges**.

You will be receiving lines until you receive the command **"Tournament"**. Each line will carry information about a pokemon and the trainer who caught it in the format:

"{trainerName} {pokemonName} {pokemonElement} {pokemonHealth}"

TrainerName is the name of the Trainer who caught the pokemon. Trainers' names are **unique**. After receiving the command **"Tournament"**, you will start receiving commands until the **"End"** command is received. They can contain one of the following:

- **"Fire"**
- **"Water"**
- **"Electricity"**

For every command, you must check if a trainer has at least 1 pokemon with the given element. If he does, he receives 1 badge. Otherwise, all of his pokemon **lose 10 health**. If a pokemon falls **to 0 or less health**, **he dies** and must be deleted from the trainer's collection. In the end, you should print all of the trainers, **sorted by the number of badges they have in descending order** (if two trainers have the same amount of badges, they should be sorted by order of appearance in the input) in the format:

"{trainerName} {badges} {numberOfPokemon}"

Examples

Input	Output

`Peter Charizard Fire 100`
`George Squirtle Water 38`
`Peter Pikachu Electricity 10`
`Tournament`
`Fire`
`Electricity`
`End`
	`Peter 2 2`
	`George 0 1`

`Sam Blastoise Water 18`
`Narry Pikachu Electricity 22`
`John Kadabra Psychic 90`
`Tournament`
`Fire`
`Electricity`
`Fire`
`End`
	`Narry 1 1`
	`Sam 0 0`
	`John 0 1`