Exercises: Object and Classes

Problems for exercise for the "PHP Fundamentals" course @ SoftUni.

You can check your solutions in Judge.

1. Order by Age

You will receive an **unknown** number of lines. On each line, you will receive array with **3** elements. **The first** element will be a **string** and represents the name of the person. **The second** element will be a **string** and will represent the **ID** of the person. **The last** element will be an **integer** and represents the **age** of the person.

When you receive the command "**End**", stop receiving input and print **all the people**, **ordered** by **age**.

Examples

Input	Output

`Georgi 123456 20`
`Pesho 78911 15`
`Stefan 524244 10`
`End`
	`Stefan with ID: 524244 is 10 years old.`
	`Pesho with ID: 78911 is 15 years old.`
	`Georgi with ID: 123456 is 20 years old.`

2. Average Grades

Define a class **`Student`**, which holds the following information about students: **name**, **list of grades** and **average grade** (calculated property, read-only). A single grade will be in range [2…6], e.g. 3.25 or 5.50.

Read an **array of students** and print the students that have **average grade ≥ 5.00** ordered **by name** (**ascending**), then by **average grade** (**descending**). Print the student name and the calculated average grade.

Examples

Input	Output

`3`
`Ivan 3`
`Todor 5 5 6`
`Diana 6 5.50`
	`Diana -> 5.75`
	`Todor -> 5.33`

`6`
`Petar 3 5 4 3 2 5 6 2 6`
`Mitko 6 6 5 6 5 6`
`Gosho 6 6 6 6 6 6`
`Ani 6 5 6 5 6 5 6 5`
`Iva 4 5 4 3 4 5 2 2 4`
`Ani 5.50 5.25 6.00`
	`Ani -> 5.58`
	`Ani -> 5.50`
	`Gosho -> 6.00`
	`Mitko -> 5.67`

3. Book Library

To model a **book library**, define classes to hold a **book** and a **library**.

The library must have a **name** and a **list of books**. The books must contain the **title**, **author**, **publisher**, **release date** (in **`dd.MM.yyyy`** format), **ISBN-number** and **price.**

Read a number **n**, followed by **n** lines of **lists of books**, add them to the library and print the **total sum of prices by author**, ordered **descending by price** and **then by author's name lexicographically**.

Books in the input will be in format **{title} {author} {publisher} {release date} {ISBN} {price}**.

The total prices must be printed **formatted to the second decimal place**.

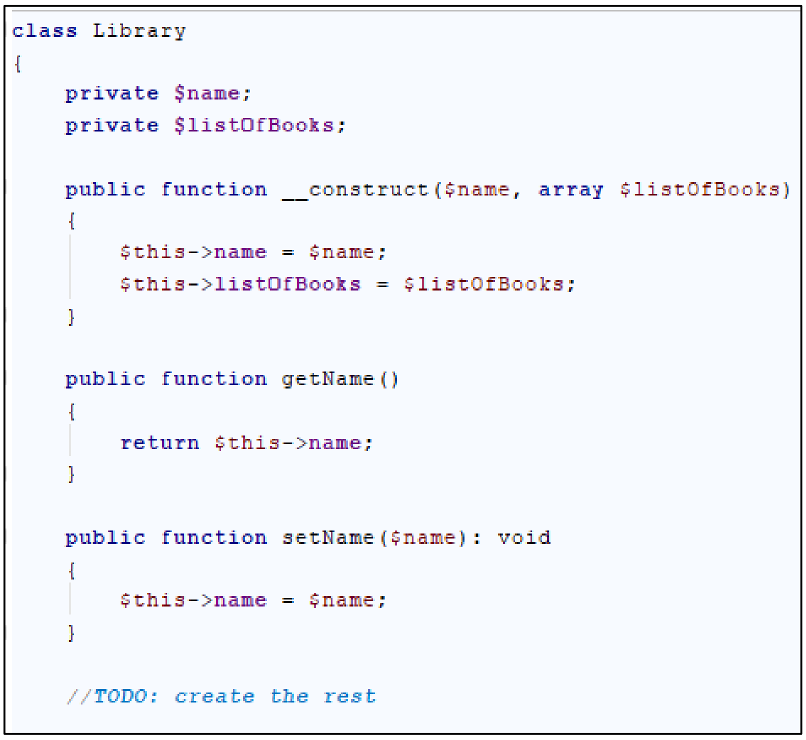
Examples

Input	Output

`5`
`LOTR Tolkien GeorgeAllen 29.07.1954 0395082999 30.00`
`Hobbit Tolkien GeorgeAll 21.09.1937 0395082888 10.25`
`HP1 JKRowling Bloomsbury 26.06.1997 0395082777 15.50`
`HP7 JKRowling Bloomsbury 21.07.2007 0395082666 20.00`
`AC OBowden PenguinBooks 20.11.2009 0395082555 14.00`
	`Tolkien -> 40.25`
	`JKRowling -> 35.50`
	`OBowden -> 14.00`

Hints

- Create classes **Book** and **`Library`** with all the mentioned above properties:



```
class Library
{
    private $name;
    private $listOfBooks;

    public function __construct($name, array $listOfBooks)
    {
        $this->name = $name;
        $this->listOfBooks = $listOfBooks;
    }

    public function getName()
    {
        return $this->name;
    }

    public function setName($name): void
    {
        $this->name = $name;
    }

    //TODO: create the rest
```

-

```
class Book
{
    private $title;
    private $author;
    private $publisher;
    private $releaseDate;
    private $Isbn;
    private $price;

    public function __construct($title, $author, $publisher, $releaseDate, $Isbn, $price)
    {
        $this->title = $title;
        $this->author = $author;
        $this->publisher = $publisher;
        $this->releaseDate = $releaseDate;
        $this->Isbn = $Isbn;
        $this->price = $price;

    }

    public function getTitle()
    {
        return $this->title;
    }

    public function setTitle($title): void
    {
        $this->title = $title;
    }

    //TODO: create the rest
```

- **Create** an object of type **`Library`**.
- **Read the input** and create a **Book** object for each book in the input.

4. Articles

Create an article class 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 **`__toString()`** – print the article in the following format: **"{title} - {content}: {author}"**
  Hint: you can check here **for more details**

Write a program that reads an article in the following format **"{title}, {content}, {author}"**. On the next line, you will get a number **n**. On the next **n lines,** you will get one of the following commands:

```
"Edit: {new content}"

"ChangeAuthor: {new author}"

"Rename: {new title}"
```

At the end, print the final 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`

5. Articles 2.0

Change the program, so you can store a **list of articles**. You will not need the methods any more (**except the `__toString()` method**). On the **first line**, you will get a number **n**. On the **next n lines**, you will get some **articles in the same format** as the previous task (**"{title}, {content}, {author}"**). Finally, you will get one of the **three inputs**:

"title"
"content"
"author"

You need to **order the articles** alphabetically based on the command and **print them sorted by the given criteria**.

Example

Input	Output

`2`
`Science, planets, Bill`
`Article, content, Johnny`
`title`
	`Article - content: Johnny`
	`Science - planets: Bill`

`3`
`title1, C, author1`
`title2, B, author2`
`title3, A, author3`
`content`
	`title3 - A: author3`
	`title2 - B: author2`
	`title1 - C: author1`

6. Mentor Group

You are mentor of a group. You have done your job well and now you have to generate a report about your group's activity. You will be given usernames and dates ("***dd/MM/yyyy***"), dates (**if any**) are separated with comma, until you receive command "**end of dates**". After that you will receive user and some comment (separated with dash). You can add comment for every user who is **in** your group (if **not** ignore the line). Adding comment/date to same user more than once should **append** to that user the comment/date. Upon receiving command "**end of comments**" you should generate report in format:

```
{user}
Comments:
- {firstComment}
…
Dates attended:
-- {firstDate}
-- {secondDate}
```

Users should be printed ordered by name (ascending). For every user, their dates should be sorted again in ascending order. Input will be valid and in the format described - you should **not check** it explicitly!

Examples

Input	Output

`nakov 22/08/2016,20/08/2016`
`simeon10 22/08/2016`
`end of dates`
`nakov-Excellent algorithmic thinking.`
`Gesh4o-Total noob.`
`end of comments`
	`nakov`
	`Comments:`
	`- Excellent algorithmic thinking.`
	`Dates attended:`
	`-- 20/08/2016`
	`-- 22/08/2016`
	`simeon10`
	`Comments:`
	`Dates attended:`
	`-- 22/08/2016`

Comments	
Not that simeon10 has no comments but we still leave the comments section. User Gesh4o does not have attendancy date so he is not registered in your group therefore he is not a part from the report.	

Input	Output

`student1`
`student2 13/11/2011`
`end of dates`
`student1-Bad.`
`student1-Good.`
`student1-Definetely good.`
`end of comments`
	`student1`
	`Comments:`
	`- Bad.`
	`- Good.`
	`- Definitely good.`
	`Dates attended:`
	`student2`
	`Comments:`
	`Dates attended:`
	`-- 13/11/2011`

7. Teamwork Projects

It's time for teamwork projects and you are responsible for making 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 that team. For every newly created team you should **print** a message:

"Team {team Name} has been created by {user}!"

Next you will receive user with 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 user tries to **create** a team more than once a message should be displayed:

- "**Team {teamName} was already created!**"

- Creator of a team cannot **create** another team - message should be thrown:
  - "**{user} cannot create another team!**"
- If user tries to **join** currently non-existing team a message should be displayed:
  - "**Team {teamName} does not exist!**"
- Member of a team cannot **join** another team - message should be thrown:
  - "**Member {user} cannot join team {team Name}!**"
- In the **end** (*after teams' report*) teams with **zero** members (with **only a creator**) should **disband**. Every **valid** team should be printed ordered by **name** (ascending) in this format:

```
"{teamName}:
- {creator}
--
{member}…"
```

Examples

Input	Output	Comments
2 Didi-PowerPuffsCoders Toni-Toni is the best Petq->PowerPuffsCoders Toni->Toni is the best end of assignment	Team PowerPuffsCoders has been created by Didi! Team Toni is the best has been created by Toni! Member Toni cannot join team Toni is the best! PowerPuffsCoders - Didi -- Petq Teams to disband: Toni is the best	Toni created a team in which he tried later to join. So message was shown. Since there is no one other who is trying to join his team the team have to **disband**.
3 Tatyana-CloneClub Helena-CloneClub Trifon-AiNaBira Pesho->aiNaBira Pesho->AiNaBira Tatyana->Leda PeshO->AiNaBira Cossima->CloneClub end of assignment	Team CloneClub has been created by Tatyana! Team CloneClub was already created! Team AiNaBira has been created by Trifon! Team aiNaBira does not exist! Team Leda does not exist! AiNaBira - Trifon -- Pesho -- PeshO CloneClub - Tatyana -- Cossima Teams to disband:	Note that when you join a team you should check **first** if it exists, **then** check if the user is already in a team: Tatyana has created CloneClub, then she tries to join a non-existing team – so message for non-existing team is shown.