**Exercise: Objects and Classes**

Problems for exercise and homework for the ["C# Fundamentals" course @ SoftUni](https://softuni.bg/modules/57/tech-module-4-0)  
You can check your solutions in [Judge](https://judge.softuni.bg/Contests/1215)

* **Advertisement Message**

Write a program that **generates random fake advertisement message** to advertise a product. The messages must consist of 4 parts: **phrase** + **event** + **author** + **city**. Use the following predefined parts:

* **Phrases** – {"Excellent product.", "Such a great product.", "I always use that product.", "Best product of its category.", "Exceptional product.", "I can’t live without this product."}
* **Events** – {"Now I feel good.", "I have succeeded with this product.", "Makes miracles. I am happy of the results!", "I cannot believe but now I feel awesome.", "Try it yourself, I am very satisfied.", "I feel great!"}
* **Authors** – {"Diana", "Petya", "Stella", "Elena", "Katya", "Iva", "Annie", "Eva"}
* **Cities** – {"Burgas", "Sofia", "Plovdiv", "Varna", "Ruse"}

The format of the output message is the following: **{phrase} {event} {author} – {city}**.

You will receive the **number of messages** to be generated. Print each random message at a separate line.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3 | Such a great product. Now I feel good. Elena – Ruse  Excellent product. Makes miracles. I am happy of the results! Katya – Varna  Best product of its category. That makes miracles. Eva - Sofia |

* **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}: {autor}"**

Write 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}"**. At 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 |

* **Articles 2.0**

Change the program in such a way, that you will be able to store a **list of articles**. You will not need to use the previous methods any more (**except the ToString method**). On the **first line**, you will receive the number of articles. On the **next lines**, you will receive the **articles in the same format** as in the previous problem: **"{title}, {content}, {author}"**. Finally, you will receive a string: **"title", "content"** or an **"author"**. You need to **order the articles** alphabetically, based on **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 |

* **Students**

Write a program that receives a **count of students - n** and **orders them by grade** in **descending order**. Each student should have a **First name** (string), a **Last name** (string) and a **grade** (a floating-point number).

**Input**

* On the first line, you are going to receive **n - the count of students**
* On the next **n** lines, you will be receiving the info about the students in the following format:

**"{first name} {second name} {grade}"**

**Output**

* Print 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 |

* **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 an 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 an 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 an 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 toprint 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  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, 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  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 a user joins a team, you should **first** check if the team exists and **then** check if the user is already in a team:  Tatyana has created CloneClub, then she tried to join a non-existent team and the concrete message was displayed. |

* **Vehicle Catalogue**

You have to create a vehicle catalogue. You will store only two types of vehicles – a **car** and a **truck**. Until you receive the “**End**” command you will be receiving **lines** of **input** in the following format:

|  |
| --- |
| **{typeOfVehicle} {model} {color} {horsepower}** |

After the “**End**” command, you will start receiving **models** of **vehicles**. Print the **data** for every received vehicle in the following format:

|  |
| --- |
| **Type: {typeOfVehicle}**  **Model: {modelOfVehicle}**  **Color: {colorOfVehicle}**  **Horsepower: {horsepowerOfVehicle}** |

When you receive the command “**Close the Catalogue**”, print the **average** **horsepower** for the **cars** and for the **trucks** in the following format:

**{typeOfVehicles} have average horsepower of {averageHorsepower}.**

The **average** **horsepower** is calculated by **dividing** the **sum** of the **horsepower** of **all** vehicles from the certain type by the **total** **count** of **vehicles** from the **same** **type**. Round the answer to the **2nd digit after the decimal separator**.

**Constraints**

* The type of vehicle will always be either a **car** or a **truck**.
* You will not receive the **same** **model** **twice**.
* The received horsepower will be an integer in the range **[1…1000]**
* You will receive at most **50** vehicles.
* The separator will always be a single **whitespace**.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| truck Man red 200  truck Mercedes blue 300  car Ford green 120  car Ferrari red 550  car Lamborghini orange 570  End  Ferrari  Ford  Man  Close the Catalogue | Type: Car  Model: Ferrari  Color: red  Horsepower: 550  Type: Car  Model: Ford  Color: green  Horsepower: 120  Type: Truck  Model: Man  Color: red  Horsepower: 200  Cars have average horsepower of: 413.33.  Trucks have average horsepower of: 250.00. |

* **Order by Age**

You will receive an **unknown** number of lines. Each line will be consisted of an array of **3** elements. **The first** element will be a string and it will represent the name of a person. **The second** element will be a **string** and it will represent the **ID** of the person. **The last** element will be an **integer** - the **age** of the person. When you receive the command "**End**", 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. |