**Exercises: Sets and Dictionaries Advanced**

Problems for exercises and homework for the ["C# Advanced" course @ HYPERLINK "https://softuni.bg/courses/csharp-advanced"SoftUni](https://softuni.bg/courses/csharp-advanced).

You can check your solutions here: <https://judge.softuni.bg/Contests/1466/Sets-and-Dictionaries-Advanced-Exercise>

* **Unique Usernames**

Write a program that reads from the console a sequence of **N usernames** and keeps a collection only of the **unique** ones.

On the **first** line you will be given an integer **N**. On the next **N** lines you will receive **one** username **per** **line**. Print the collection on the console in **order** of **insertion**:

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 6  John  John  John  Peter  John  Boy1234 | John  Peter  Boy1234 |
| 10  Peter  Maria  Peter  George  Sam  Maria  Sara  Peter  Sam  George | Peter  Maria  George  Sam  Sara |

* **Sets of Elements**

Write a program that prints a **set of elements**.

On the first line you will receive two numbers - **n** and **m**, which represent the lengths of two separate sets.

On the next **n** + **m** lines you will receive **n** numbers, which are the numbers in the **first** set, and **m** numbers, which are in the **second** set.

Find all the **unique** **elements** that appear in **both of them** and **print** them in the order in which they appear in the **first** set - **n**.

**For example:**

Set with length n = 4: {1, **3**, **5**, 7}

Set with length m = 3: {**3**, 4, **5**}

Set that contains all the **elements** that repeat in **both** **sets** -> {**3**, **5**}

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 4 3  1  3  5  7  3  4  5 | 3 5 |
| 2 2  1  3  1  5 | 1 |

* **Periodic Table**

Write a program that keeps all the **unique** chemical **elements**.

On the first line you will be given a number **n** - the **count** of input **lines** that you are going to receive. On the next **n** lines you will be receiving **chemical** **compounds**, separated by a **single** **space**. Your task is to print all the **unique ones** in **ascending** **order**:

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 4  Ce O  Mo O Ce  Ee  Mo | Ce Ee Mo O |
| 3  Ge Ch O Ne  Nb Mo Tc  O Ne | Ch Ge Mo Nb Ne O Tc |

* **Even Times**

Write a program that **prints** a **number** from a collection, which appears an **even** **number** of **times** in it.

On the first line, you will be given **n** – the **count** of **integers** you will receive.

On the next n lines you will be receiving **the numbers**. It is **guaranteed** that **only** **one** of them **appears** an **even** **number** of times.

Your task is to **find** that **number** and **print** it in the end.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  2  -1  2 | 2 |
| 5  1  2  3  1  5 | 1 |

* **Count Symbols**

Write a program that reads some **text** from the console and **counts** the **occurrences** of **each** character in it.

Print the results in **alphabetical** (lexicographical) order.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| SoftUni rocks | : 1 time/s  S: 1 time/s  U: 1 time/s  c: 1 time/s  f: 1 time/s  i: 1 time/s  k: 1 time/s  n: 1 time/s  o: 2 time/s  r: 1 time/s  s: 1 time/s  t: 1 time/s |
| Did you know Math.Round rounds to the nearest even integer? | : 9 time/s  .: 1 time/s  ?: 1 time/s  D: 1 time/s  M: 1 time/s  R: 1 time/s  a: 2 time/s  d: 3 time/s  e: 7 time/s  g: 1 time/s  h: 2 time/s  i: 2 time/s  k: 1 time/s  n: 6 time/s  o: 5 time/s  r: 3 time/s  s: 2 time/s  t: 5 time/s  u: 3 time/s  v: 1 time/s  w: 1 time/s  y: 1 time/s |

* **Wardrobe**

Write a program that helps you decide what **clothes** to wear from your **wardrobe**.

You will receive the **clothes**, which are currently in your wardrobe, sorted by their **color** in the following **format**:

"**{color} -> {item1},{item2},{item3}…**"

If you receive a certain color, which already **exists** in your wardrobe, just **add** the clothes to **its** **records**.

You can also receive **repeating** **items** for a certain **color** and you have to keep their **count**.

In the end, you will receive a **color** and a piece of **clothing**, which you will **look for** in the wardrobe, separated by a space in the following format:

**"{color} {clothing}"**

Your task is to print all the **items** and their **count** for **each** **color** in the following format**:**

**"{color}** **clothes**:

\* **{item1}** - **{count}**

\* **{item2}** - **{count}**

\* **{item3}** - **{count}**

…

\* **{itemN}** - **{count}"**

If you find the **item** you are **looking for**, you need to print **"(found!)"** next to it:

"\* **{itemN}** - **{count} (found!)"**

**Input**

* On the **first** **line**, you will receive **n** – the **number of lines** of clothes, which you will receive.
* On the next **n** lines, you will receive the **clothes** in the **format** **described** above.

**Output**

* Print the **clothes** from your wardrobe in the **format** **described** above

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 4  Blue -> dress,jeans,hat  Gold -> dress,t-shirt,boxers  White -> briefs,tanktop  Blue -> gloves  Blue dress | Blue clothes:  \* dress - 1 (found!)  \* jeans - 1  \* hat - 1  \* gloves - 1  Gold clothes:  \* dress - 1  \* t-shirt - 1  \* boxers - 1  White clothes:  \* briefs - 1  \* tanktop - 1 |
| 4  Red -> hat  Red -> dress,t-shirt,boxers  White -> briefs,tanktop  Blue -> gloves  White tanktop | Red clothes:  \* hat - 1  \* dress - 1  \* t-shirt - 1  \* boxers - 1  White clothes:  \* briefs - 1  \* tanktop - 1 (found!)  Blue clothes:  \* gloves - 1 |
| 5  Blue -> shoes  Blue -> shoes,shoes,shoes  Blue -> shoes,shoes  Blue -> shoes  Blue -> shoes,shoes  Red tanktop | Blue clothes:  \* shoes - 9 |

* **\*The V-Logger**

Create a program that keeps information about **vloggers** and their **followers**.

The **input** will come as a sequence of strings, where each string will represent a **valid** command. The commands will be presented in the following format:

* "**{vloggername}**" **joined The V-Logger** – keep the vlogger in your records.
* Vloggernames **consist** **of only one word**.
* If the **given** **vloggername** already **exists, ignore** thatcommand**.**
* "**{vloggername} followed {vloggername}**" – The first vlogger followed the second vlogger.
* If **any** of the **given vlogernames** **does not exist** in you collection, **ignore** thatcommand**.**
* Vlogger **cannot** follow **himself**
* Vlogger **cannot** follow someone he is **already a follower** **of**
* **"Statistics" -** Upon receiving this command, you have to print a statistic about the vloggers.

Each vlogger has an unique **vloggername**.

**Vloggers** can **follow other vloggers** and a vlogger **can follow** **as many other vloggers** **as he wants**, but he **cannot** follow **himself** or follow someone he is **already a follower** **of**.

You need to print the **total** **count** of **vloggers** in your collection.

Then you have to print the **most** **famous** **vlogger** – the one with the most followers, with **his** **followers.** If more than one vloggers have the **same** **number** of **followers**, print theone **following less** people and his **followers** should be printed in **lexicographical order (**in case the vlogger has **no followers**, print just the first line, which is described **below**).

Lastly, print the **rest** **vloggers**, ordered by the **count** of followers in **descending** order, then by the number of vloggers he follows in **ascending** **order.**

The **whole output must be** in the following format:

**"The V-Logger has a total of {registered vloggers} vloggers in its logs.**

**1. {mostFamousVlogger} : {followers} followers, {followings} following**

**\* {follower1}**

**\* {follower2} …**

**{No}. {vlogger} : {followers} followers, {followings} following**

**{No}. {vlogger} : {followers} followers, {followings} following…"**

**Input**

* The input will come in the format described above.

**Output**

* On the first line, print **the total count of vloggers** in the format described above.
* On the second line, print the **most** **famous** vlogger in the format described above.
* On the **next** lines, print all of the **rest** vloggers in the format described above.

**Constraints**

* There will be **no** **invalid** input.
* There will be no situation where **two** **vloggers** have **equal** count of **followers** and **equal** count of **followings**
* Allowed time/memory: **100ms/16MB**.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| EmilConrad joined The V-Logger  VenomTheDoctor joined The V-Logger  Saffrona joined The V-Logger  Saffrona **followed** EmilConrad  Saffrona **followed** VenomTheDoctor  EmilConrad **followed** VenomTheDoctor  VenomTheDoctor **followed** VenomTheDoctor  Saffrona **followed** EmilConrad  Statistics | The V-Logger has a total of 3 vloggers in its logs.  1. VenomTheDoctor : 2 followers, 0 following  \* EmilConrad  \* Saffrona  2. EmilConrad : 1 followers, 1 following  3. Saffrona : 0 followers, 2 following |
| JennaMarbles joined The V-Logger  JennaMarbles followed Zoella  AmazingPhil joined The V-Logger  JennaMarbles followed AmazingPhil  Zoella joined The V-Logger  JennaMarbles followed Zoella  Zoella followed AmazingPhil  Christy followed Zoella  Zoella followed Christy  JacksGap joined The V-Logger  JacksGap followed JennaMarbles  PewDiePie joined The V-Logger  Zoella joined The V-Logger  Statistics | The V-Logger has a total of 5 vloggers in its logs.  1. AmazingPhil : 2 followers, 0 following  \* JennaMarbles  \* Zoella  2. Zoella : 1 followers, 1 following  3. JennaMarbles : 1 followers, 2 following  4. PewDiePie : 0 followers, 0 following  5. JacksGap : 0 followers, 1 following |

* **\*Ranking**

Write a program that **ranks** candidate-interns, depending on the **points** from the **interview** **tasks** and their **exam** **results** in SoftUni.

You will receive some lines of **input** in the format **"{contest}:{password for contest}"** until you receive **"end of contests"**.

Save that data because **you will need it later**.

After that you will receive other type of inputs in format **"{contest}=>{password}=>{username}=>{points}"** until you receive **"end of submissions"**. Here is what you need to do:

* Check if the **contest is valid (if you received it in the first type of input)**
* Check if the **password is correct for the given contest**
* Save the user with the **contest** they take part in **(a user can take part in many contests)** and the points the user has in the **given** **contest**. If you receive the **same contest** andthe **same user, update** the **points only if the new ones** are **more than** the **older ones.**

At the end you have to print the info for the user with the **most points** in the format:

**"Best candidate is {user} with total {total points} points."**.

After that print **all students ordered** bytheir **names**.

For **each** user,print **each contest** withthe **points** in **descending** order in the following format:

**"{user1 name}**

**# {contest1} -> {points}**

**# {contest2} -> {points}**

**{user2 name}**

**…"**

**Input**

* You will be receiving strings in formats described above, until the appropriate commands, also described above, are given.

**Output**

* On the **first** line print the best user in the format **described** above.
* On the **next** lines print all students ordered as mentioned above in format.

**Constraints**

* There will be **no** case with two **equal** **contests**.
* The **strings** may contain any ASCII character except from **(:, =, >).**
* The **numbers** will be in range **[0 - 10000]**.
* The **second** input is always **valid**.
* There will be no case with **2** or **more** users with **same** **total** **points**.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| Part One Interview:success  Js Fundamentals:Peter  C# Fundamentals:fundPass  Algorithms:fun  end of contests  C# Fundamentals=>fundPass=>Tanya=>350  Algorithms=>fun=>Tanya=>380  Part One Interview=>success=>Nikola=>120  Java Basics Exam=>pesho=>PeterPeter=>400  Part One Interview=>success=>Tanya=>220  OOP Advanced=>password123=>JohnJohn=>231  C# Fundamentals=>fundPass=>Tanya=>250  C# Fundamentals=>fundPass=>Nikola=>200  Js Fundamentals=>Peter=>Tanya=>400  end of submissions | Best candidate is Tanya with total 1350 points.  Ranking:  Nikola  # C# Fundamentals -> 200  # Part One Interview -> 120  Tanya  # Js Fundamentals -> 400  # Algorithms -> 380  # C# Fundamentals -> 350  # Part One Interview -> 220 |
| Java Advanced:funpass  Part Two Interview:success  Math Concept:asdasd  Java Web Basics:forrF  end of contests  Math Concept=>ispass=>Monika=>290  Java Advanced=>funpass=>Simona=>400  Part Two Interview=>success=>Dany=>120  Java Advanced=>funpass=>Peter=>90  Java Web Basics=>forrF=>Simona=>280  Part Two Interview=>success=>Peter=>0  Math Concept=>asdasd=>Dany=>250  Part Two Interview=>success=>Simona=>200  end of submissions | Best candidate is Simona with total 880 points.  Ranking:  Dany  # Math Concept -> 250  # Part Two Interview -> 120  Peter  # Java Advanced -> 90  # Part Two Interview -> 0  Simona  # Java Advanced -> 400  # Java Web Basics -> 280  # Part Two Interview -> 200 |