**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 |