# <https://judge.softuni.bg/Contests/1578/Exam-Preparation-ll>

# Problem 1. Date estimation

## Input / Constraints

Today is your exam. **It’s 26th of** **August 2018**. you will be given a single date in **format year-month-day**. You should estimate if the date has **passed** regarding to the date mention above (2018-08-26), **if it is not or if it is today. If it is not** you should print how many days are left till that date. **Note that the current day stills count!**

**Date Format:  
yyyy-mm-dd**

## Output

The output should be printed on the console.

If the date has **passed** you should print the following output:

* **"**Passed**"**

If the day is **today**:

**"**Today date**"**

If the date is **future**:

* **"**{**number of days**} days**"**

## Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| INPUT  2018-08-20 | Passed | Because 20 august is before 26 august we print that the day is Passed |
| **Input** | **Output** | **Comments** |
| 2021-08-26 | 1097 days left | We estimate that the 26th of august 2021 is after 1096 and beacause 26 stills count we add 1 => 1097 |

# Problem 2. Lists

## Input / Constraints

You will be given a **single lines of elements**(**integers**), **separated** with **one or more spaces**. You should check if **all** **elements in the line are unique**. If they are you should increase the value of every even element with the number of 2 and print the list on single row in ascending order separated by ",".

If they are not unique you should increase every odd element with the number of 3 and print them on single row, separated with ":"

On the next line you should print sum of the all elements divided by the count of the elements in the list. You should do that until you receive the command "stop playing"

## Output

If the elements are unique  
Unique list: {elements in the list, separated by “,”}  
Output: {sum of all elements divided by the length of the list}  
Else

Non-unique list: {elements in the list, separated by “:”}

Output: {sum of all elements divided by the length of the list}

## Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 1 2 3 4 5 6  1 1 2 2 1 4 7 7 8 8  5 5 5 5  stop playing | Unique list: 1,3,4,5,6,8  Output: 4.50  Non-unique list: 2:2:4:4:4:4:8:8:10:10  Output: 5.60  Non-unique list: 8:8:8:8  Output: 8.00 | First is unique so we add to every even elemnt 2. \the list looks like this: 1, 4, 3, 6,5,8  After that we order them by ascending and the same list looks like this: 1,3,4,5,6,8  Output = 1+3+4+5+6+8 = 27/6 = 4.50  The elements in the list are not unique, so we add to every odd element 3. [\\\\\The](file:///\\The) list looks like this: 3:3:2:2:3:1:10:10:8:8.We order them by ascending and it becomes: 2:2:4:4:4:4:8:8:10:10  Output 56/10 = 5.60  The elements are not unique so we add to every odd eleme:nt 3 and becomes like this: 8:8:8:8  Output: 32/4 = 8 |
| **Input** | **Output** | **Comments** |
| 1 1 1  stop playing | Non-unique list: 4:4:4  Output: 4.00 |  |

# Problem 4 – Arena Tier

Pesho is a pro gladiator, he is struggling to become master of the Arena. // TODO some more story

You will receive **several input lines** in one of the following formats:

"{gladiator} -> {technique} -> {skill}"

"{gladiator} vs {gladiator}"

The gladiator and technique are strings, the given **skill** will be an integer number. You need to keep track of **every gladiator**.

When you receive a **gladiator and his technique and skill**, add him to the gladiator pool, if he isn`t present, else add his technique or update his skill, only if the current technique skill is lower than the new value.

If you receive **"{gladiator} vs {gladiator}"** and both gladiators exist in the tier, they duel with the following rules:

Compare their techniques, if they got at least one in common, the gladiator with better total skill points wins and the other is demoted from the tier -> remove him.

If they don`t have techniques in common, the duel isn`t happening and both continue in the Season.

You should end your program when you receive the command "Ave Cesar". At that point you should print the gladiators, **ordered by total skill in desecending order, then ordered by name in ascending order**. Foreach gladiator print their technique and skill, **ordered desecending, then ordered by technique name in ascending order**

## Input / Constraints

You will receive input on several lines.

* The input comes in the form of commands in one of the formats specified above.
* Gladiator and technique **will always be one word string, containing no whitespaces**.
* Skill will be an **integer** in the **range [0, 1000]**.
* There will be **no invalid** input lines.
* The programm ends when you receive the command "Ave Cesar".

## Output

* The output format for each gladiator is:

"{gladiator}: {totalSkill} skill"

"- {technique} <!> {skill}"

***Scroll down to see examples.***

## Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| Pesho -> BattleCry -> 400  Gosho -> PowerPunch -> 300  Stamat -> Duck -> 200  Stamat -> Tiger -> 250  Ave Cesar | Stamat: 450 skill  - Tiger <!> 250  - Duck <!> 200  Pesho: 400 skill  - BattleCry <!> 400  Gosho: 300 skill  - PowerPunch <!> 300 | We order the gladiators by total skill points descending, then by name. We print every technique along its skill ordered descending by skill, then by technique name. |
| **Input** | **Output** |  |
| Pesho -> Duck -> 400  Julius -> Shield -> 150  Gladius -> Heal -> 200  Gladius -> Support -> 250  Gladius -> Shield -> 250  Pesho vs Gladius  Gladius vs Julius  Gladius vs Gosho  Ave Cesar | Gladius: 700 skill  - Support <!> 250  - Shield <!> 250  - Heal <!> 200  Pesho: 400 skill  - Duck <!> 400 | Gladius and Pesho don`t have common technique, so the duel isn`t valid.  Gladius wins vs Julius /common technique: "Shield". Julius is demoted.  Gosho doesn`t exist so the duel isn`t valid.  We print every gladiator left in the tier. |

# Problem 4. Train System

The **Bulgarian Dreadful Zug** Company (**BDZ**) just upgraded their ticket systems from something running on DOS 5 to a state-of-the-art system running a fancy web app. Of course, nobody bothered to migrate all the discount cards over to the new system, so all the passengers trying to buy a ticket with a discount nearly miss their train, since the cashier, grandma Penka, needs to enter it into the new system. Luckily, she heard about this great thing, called Python from her neighbor, grandma Gina, and how she can use it to automate this task.

In this problem, you assume the role of grandma Penka. You have a 15-minute lunch break, so you decide to use that time to write up a Python script to migrate all the data.

## Input / Constraints

* On the **first line**, you will receive the number of **existing cards** **N** – **integer** in **range [0-5]**
* On the next **N** lines, you will receive the **existing cards** in the format “firstName lastName ticketNum”.
  + Existing cards will **always** have a **valid card number**
* Then, until you receive the command “time to leave!”, keep reading lines in the format:
  + “createTicket firstName lastName destination cardNumber”

The **input data will always be in the format specified**. **There is no need to check it explicitly**.

When you receive all the **existing cards**, insert them into the system. Cards are tied to the person’s **full name** (**first name + space (“ “) + last name**). **A person** can have **multiple cards**.

After that, your lunch break is over and you need to start **issuing tickets** to people again. A standard “**issue ticket**” command looks like this:

* “createTicket firstName lastName destination cardNumber”

You need to check if this person **already** has a **card** with that number under that name. If they do, issue their ticket with the card number. If not, you need to check if the **card number** is valid. A **valid card number**’s **digit sum** is **divisible by 7** (example: **297296** 🡺 (**2+9+7+2+9+6**) **% 7 == 0** 🡺 **valid**). If the **card number** is **invalid**, print “card {cardNumber} is not valid!” and issue the ticket **without a discount**.

If the card number is **valid**, but it **already belongs** **to another passenger**, print “card {cardNumber} already exists for another passenger!” and issue the ticket **without a discount**.

If the card **doesn’t already belong** to **another passenger** and is **not already** in the **existing cards**, you need to **issue** that passenger a card. **Insert** it into the **cards** and print (“issuing card {cardNumber}”). After that, issue them a ticket **with a discount**.

The **price** of the **ticket** is the **sum of the ASCII codes** of the **destination name**, **divided** by **100**

* Example: **vidin** 🡺 **v+i+d+i+n** 🡺 **538 / 100** 🡺 **5.38**

Every ticket bought with a **valid card** that **belongs to the passenger** is **50% cheaper**.

## Output

When you receive “time to leave!”, print all passengers in the following format:

|  |
| --- |
| **fullName:**  **--{destination}: {ticketPrice:.2f}**  **--{destination}: {ticketPrice:.2f} (using card {cardNumber})**  **...**  **total: {ticketPrice:.2f}BGN** |

Sort the **passengers** by the **sum of their ticket prices** (**descending**). Sort **each passenger’s** **tickets** by the **ticket’s price** (**descending**). If a ticket was **bought** **without a discount**, **don’t print** “using card…” after it.

## Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 4  pesho ivanov 297296  ivan tsekov 652530  gosho goshov 547989  ivan tsekov 468845  createTicket pesho ivanov vidin 297296  createTicket ivan petrov ruska\_bela 590432  createTicket ivan petrov sofia 590430  createTicket pesho ivanov boychinovtsi 590554  createTicket bay ivan montana 912839  time to leave! | card 590432 is not valid!  issuing card 590430  issuing card 590554  card 912839 is not valid!  ivan petrov:  --ruska\_bela: 10.49lv  --sofia: 2.65lv (using card 590430)  total: 13.14lv  pesho ivanov:  --boychinovtsi: 6.57lv (using card 590554)  --vidin: 2.69lv (using card 297296)  total: 9.26lv  bay ivan:  --montana: 7.50lv  total: 7.50lv |
| **Input** | **Output** |
| 1  georgi georgiev 586790  createTicket pesho petrov vidin 297296  createTicket pesho petrov montana 630534  createTicket pesho petrov plovdiv 630534  createTicket bay ivan vidin 297296  createTicket bay ivan sofia 111111  createTicket bay ivan montana 111111  createTicket bay ivan mezdra 111111  time to leave! | issuing card 297296  issuing card 630534  card 297296 already exists for another passenger!  card 111111 is not valid!  card 111111 is not valid!  card 111111 is not valid!  bay ivan:  --montana: 7.50lv  --mezdra: 6.43lv  --vidin: 5.38lv  --sofia: 5.30lv  total: 24.61lv  pesho petrov:  --plovdiv: 3.86lv (using card 630534)  --montana: 3.75lv (using card 630534)  --vidin: 2.69lv (using card 297296)  total: 10.30lv |
| **Input** | **Output** |
| 3  ivan ivanov 094859  ceko cekov 328994  krali marko 774154  createTicket ivan ivanov montana 094859  createTicket ivan ivanov vidin 094859  createTicket ivan ivanov plovdiv 094859  createTicket krali marko vidin 774154  createTicket krali marko sofia 774154  createTicket bay ivan mezdra 000006  createTicket ceko cekov montana 328994  time to leave! | card 000006 is not valid!  ivan ivanov:  --plovdiv: 3.86lv (using card 094859)  --montana: 3.75lv (using card 094859)  --vidin: 2.69lv (using card 094859)  total: 10.30lv  bay ivan:  --mezdra: 6.43lv  total: 6.43lv  krali marko:  --vidin: 2.69lv (using card 774154)  --sofia: 2.65lv (using card 774154)  total: 5.34lv  ceko cekov:  --montana: 3.75lv (using card 328994)  total: 3.75lv |