# Problem 1. National Court

*Every day thousands of people pass by the reception at "National Court" with various questions to ask and the employees have to help everyone by providing correct information and to answer all questions.*

There are **3 employees** working on the reception all day long. Each of them can handle different number of **people** **per hour**. Your task is to calculate **how much time** it will take **to** **answer all the questions** of a given number **of people**.

First you will receive 3 lines with integers, representing the **count of people** that each of the **employee can help per hour.** On the next line you will receive the **total** **people count** as a single integer.

Every **fourth hour** all the employees **have a one-hour break** before they start working again. This is the only break they get because they don`t need rest and have no personal life. Calculate the time needed to answer all people`s questions and print it in the following format: "Time needed: {time}h."

### Input / Constraints

* On first three lines - **each employee`s efficiency** - an integer in the range **[1 - 100]**
* On the fourth line - **people count** – an integer in the range **[0 – 10000]**
* Input will always be valid and in the range specified

### Output

* Print a single line: "Time needed: {time}h."
* Allowed working **time** / **memory**: **100ms** / **16MB**

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comment** |
| 5  6  4  20 | Time needed: 2h. | All employees can answer 15 people per hour. After the first hour there are 5 people left to be answered.  All people will be answered in the second hour. |
| 1  2  3  45 | Time needed: 10h. | All employess can answer 6 people per hour. In the first 3 hours they have answered 6 \* 3 = 18 people. Then they have a break for an hour.  After the next 3 hours there are  18 + 6 \* 3 = 36 answered people.  After the break for an hour, there are only 9 people to answer.  So in the 10th hour all of the people questions would be answered. |
| 3  2  5  40 | Time needed: 5h. |  |

# Problem 2. Shopping List

*It’s the end of the week and it is time for you to go shopping, so you need to create a shopping list first.*

### Input

You will receive an **initial list** with groceries separated by **"!"**.

After that you will be receiving **4 types** of commands, until you receive **"Go Shopping!"**

* **Urgent {item} -** **add** the item at the **start** of the list. If the item **already exists,** skip this command.
* **Unnecessary {item} - remove** the item with the given name, only **if it exists** in the list. Otherwise skip this command.
* **Correct {oldItem} {newItem} –** if the item with the given **old name** exists, **change** its name with the **new** one. If it **doesn't exist**, skip this command.
* **Rearrange {item} -** if the grocery exists in the list, **remove** it from its **current position** and **add** it at the **end** of the list.

### Constraints

* There won`t be any duplicate items in the initial list

### Output

Print the **list** with all the groceries, joined by **", ".**

* **"{firstGrocery}, {secondGrocery}, …{nthGrocery}"**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Tomatoes!Potatoes!Bread  Unnecessary Milk  Urgent Tomatoes  Go Shopping! | Tomatoes, Potatoes, Bread |
| **Input** | **Output** |
| Milk!Pepper!Salt!Water!Banana  Urgent Salt  Unnecessary Grapes  Correct Pepper Onion  Rearrange Grapes  Correct Tomatoes Potatoes  Go Shopping! | Milk, Onion, Salt, Water, Banana |

# Problem 3. Heart Delivery

*Valentine’s Day is coming, and Cupid has very limited time to spread some love across the neighborhood. Help him with his mission!*

You will receive a **string** with **even integers,** separated by a **"@".** This is our neighborhood. After that a series of **Jump** commands will follow, until you receive **"Love!"** Every house in the neighborhood needs a certain number of **hearts** delivered by Cupid, in order to be able to celebrate Valentine’s Day. Those needed hearts are indicated by the integers in the neighborhood.

Cupid starts at the position of the **first** **house** (index 0) and must jump by a **given length.** The jump commands will be in this format: **"Jump {length}"**.

Every time he jumps from one house to another, the needed hearts for the visited house are **decreased by 2**. If the needed hearts for a certain house become **equal to 0** , print on the console **"Place {houseIndex} has Valentine's day."** If **Cupid** jumps to a house where the needed hearts are **already** **0,** print on the console"**Place {houseIndex} already had Valentine's day.**".

Keep in mind that **Cupid** can have a **bigger jump length** than the **size of the neighborhood** and if he does jump **outside** of it, he should **start** from the **first house** again**.**

*For example, we are given this neighborhood: 6@6@6. Cupid is at the start and jumps with a length of 2. He will end up at index 2 and decrease the needed hearts there by 2: [6, 6, 4]. Next he jumps again with a length of 2 and goes outside the neighborhood, so he goes back to the first house (index 0) and again decreases the needed hearts there: [4, 6, 4].*

### Input

* On the first line you will receive a **string** with **even integers** separated by **"@"** –the neighborhood and the number of hearts for each house.
* On the next lines, until "**Love!**" is received, you will be getting jump commands in this format: "**Jump {length}**".

### Output

At the end print **Cupid's** **last position** and whether his mission was successful or not:

* "**Cupid's last position was {lastPositionIndex}.**"
* If **each house** has had a Valentine's day, print:
  + "**Mission was successful.**"
* If **not,** print the **count** of all houses that **didn`t** celebrate a Valentine's Day:
  + **"Cupid has failed {houseCount} places."**

### Constraints

* The **neighborhood`s** size will be in the range [1…20]
* Each **house** will need an **even number** of hearts in the range [2 … 10]
* Each **jump length** will be an integer in the range [1 … 20]

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 10@10@10@2  Jump 1  Jump 2  Love! | Place 3 has Valentine's day.  Cupid's last position was 3.  Cupid has failed 3 places. | Jump 1 ->> [10, 8, 10, 2]  Jump 2 ->> [10, 8, 10, 0] so we print "Place 3 has Valentine's day."  Next command is "Love!", so we print Cupid`s last position and the outcome of his mission. |
| 2@4@2  Jump 2  Jump 2  Jump 8  Jump 3  Jump 1  Love! | Place 2 has Valentine's day.  Place 0 has Valentine's day.  Place 0 already had Valentine's day.  Place 0 already had Valentine's day.  Cupid's last position was 1.  Cupid has failed 1 places. |  |