Problem 1 – The Imitation Game

A problem for exam preparation for the "C# Fundamentals" course @ SoftUni
Submit your solutions in the SoftUni Judge system here

During World War 2, you are a mathematician who has joined the cryptography team to decipher the enemy's enigma code. Your job is to create a program to crack the codes.

On the first line of the input, you will receive the **encrypted message**. After that, until the **"Decode"** command is given, **you will be receiving strings** with **instructions** for different **operations** that need to be performed upon the **concealed message** to **interpret it** and reveal its true content. There are several types of instructions, split by **'|'**

- **"Move {number of letters}"**:
  - **Moves** the **first n letters** to the **back** of the string
- **"Insert {index} {value}"**:
  - **Inserts** the given value **before the given index** in the string
- **"ChangeAll {substring} {replacement}"**:
  - **Changes all occurrences** of the given substring with the replacement text

Input / Constraints

- On the first line, you will receive a string with a message.
- On the following lines, you will be receiving commands, split by **'|'.**

Output

- After the **"Decode"** command is received, print this message:
  "The decrypted message is: {message}"

Examples

Input	Output
zzHe ChangeAll|z|l Insert|2|o Move|3 Decode	The decrypted message is: Hello
Comments	
ChangeAll|z|l zzHe → llHe (We replace all occurrences of 'z' with 'l') **Insert|2|o** llHe → lloHe (We add an 'o' before the character on index 2) Move|3	

lloHe → Hello (We take the first three characters and move them to the end of the string) Finally, after receiving the **"Decode"** command, we print the resulting message.	
Input	**Output**
owyouh Move|2 Move|3 Insert|3|are Insert|9|? Decode	The decrypted message is: howareyou?

Problem 2 – AdAstra

A problem for exam preparation for the "C# Fundamentals" course @ SoftUni
Submit your solutions in the SoftUni Judge system here

You are an astronaut who just embarked on a mission across the solar system. Since you will be in space for a long time, you have packed a lot of food with you. Create a program, which helps you identify how much food you have left and gives you information about its expiration date.

On the first line of the input, you will be given a **text string**. You must extract the information about the food **and calculate the total calories.**

First, you must **extract the food info**. It will always follow the same pattern rules:

- It will be surrounded by **"|"** or **"#"** (only one of the two) in the following pattern:
  `#{item name}#{expiration date}#{calories}#` or
  `|{item name}|{expiration date}|{calories}|`
- The item name will contain **only lowercase and uppercase letters and whitespace**
- The expiration date will always follow the pattern: **"{day}/{month}/{year}", where the day, month, and year will be exactly two digits long**
- The calories will be **an integer between 0-10000**

Calculate **the total calories of all food items** and then determine **how many days you can last with the food you have**. Keep in mind that **you need 2000kcal a day**.

Input / Constraints

- You will receive **a single string**

Output

- First, print **the number of days** you will be able to last with the food you have:
  `"You have food to last you for: {days} days!"`
  - **The output for each food item should look like this:**
    `"Item: {item name}, Best before: {expiration date}, Nutrition: {calories}"`

Examples

Input	
#Bread#19/03/21#4000#|Invalid|03/03.20||Apples|08/10/20|200||Carrots|06/08/20|500||Not right|6.8.20|5|	
Output	**Comments**
You have food to last you for: 2 days! Item: Bread, Best before: 19/03/21, Nutrition: 4000 Item: Apples, Best before: 08/10/20, Nutrition: 200 Item: Carrots, Best before: 06/08/20, Nutrition: 500	We have a total of three matches – bread, apples, and carrots. The sum of their calories is 4700. Since you need 2000kcal a day, we divide 4700/2000, which means this food will last you for 2 days. We print each item.
Input	
$$#@@%^&#Fish#24/12/20#8500#|#Incorrect#19.03.20#450|$5*(@!#Ice Cream#03/10/21#9000#^#@aswe|Milk|05/09/20|2000|	
Output	**Comments**
You have food to last you for: 9 days! Item: Fish, Best before: 24/12/20, Nutrition: 8500 Item: Ice Cream, Best before: 03/10/21, Nutrition: 9000 Item: Milk, Best before: 05/09/20, Nutrition: 2000	We have three matches. The total calories are 8500 + 9000 + 2000 = 19500, which means you have food for a total of 9 days.
Input	
Hello|#Invalid food#19/03/20#450|$5*(@	
Output	**Comments**
You have food to last you for: 0 days!	We have no matches, which means we have no food. The colored text is not a match since it doesn't have a # at the end.

Problem 3 – Plant Discovery

A problem for exam preparation for the "C# Fundamentals" course @ SoftUni

Submit your solutions in the SoftUni Judge system here

You have now returned from your world tour. On your way, you have discovered some new plants, and you want to gather some information about them and create an exhibition to see which plant is highest rated.

On the **first line,** you will receive a number **n**. On the next **n lines**, you will be given some information about the plants that you have discovered in the format: **"{plant}<->{rarity}"**. **Store** that **information** because you will need it later. If you receive a plant **more than once**, **update** its rarity.

After that, until you receive the **command "Exhibition"**, you will be given some of these **commands**:

- **"Rate: {plant} - {rating}"** – **add** the given **rating** to the plant (**store all ratings**)
- **"Update: {plant} - {new_rarity}"** – **update** the **rarity** of the plant with the **new one**
- **"Reset: {plant}"** – **remove all** the **ratings** of the given plant

Note: If any given plant name is invalid, print "error"

After the command **"Exhibition"**, print the information that you have about the plants in the following format:

```
"Plants for the exhibition:
- {plant_name1}; Rarity: {rarity}; Rating: {average_rating}
- {plant_name2}; Rarity: {rarity}; Rating: {average_rating}
…
- {plant_nameN}; Rarity: {rarity}; Rating: {average_rating}"
```

The **average rating** should be formatted to the **second decimal place.**

Input / Constraints

- You will receive the input as described above

Output

- Print the **information** about all plants as **described above**

Examples

Input	Output
3 Arnoldii<->4 Woodii<->7 Welwitschia<->2 Rate: Woodii - 10 Rate: Welwitschia - 7 Rate: Arnoldii - 3 Rate: Woodii - 5	Plants for the exhibition: - Arnoldii; Rarity: 4; Rating: 0.00 - Woodii; Rarity: 5; Rating: 7.50 - Welwitschia; Rarity: 2; Rating: 7.00

Update: Woodii - 5 Reset: Arnoldii Exhibition	
2 Candelabra<->10 Oahu<->10 Rate: Oahu - 7 Rate: Candelabra - 6 Exhibition	Plants for the exhibition: - Candelabra; Rarity: 10; Rating: 6.00 - Oahu; Rarity: 10; Rating: 7.00