Problem 1 - The Imitation Game

Problem for exam preparation for the Programming Fundamentals Course @SoftUni.
Submit your solutions in the SoftUni judge system at https://judge.softuni.org/Contests/Practice/Index/2525#0.

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	

llloHe → 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?`

JS 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',`
	`The decrypted message is: howareyou?`

`'Insert|3|are',`
`'Insert|9|?'`
`'Decode',`
`]`	

Problem 2 - Ad Astra

Problem for exam preparation for the [Programming Fundamentals Course @SoftUni](#).
Submit your solutions in the SoftUni judge system at https://judge.softuni.org/Contests/Practice/Index/2525#1.

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.

JS 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.

JavaScript 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 - The Pianist

Problem for exam preparation for the Programming Fundamentals Course @SoftUni.
Submit your solutions in the SoftUni judge system at https://judge.softuni.org/Contests/Practice/Index/2525#2.

You are a pianist, and you like to keep a list of your favorite piano pieces. Create a program to help you organize it and add, change, remove pieces from it!

On the first line of the standard input, you will receive an integer **n** – the **number of pieces** you will initially have. On the next **n** lines, the **pieces themselves** will follow with their **composer** and **key**, separated by **"|"** in the following format: **"{piece}|{composer}|{key}"**.

Then, you will be receiving different **commands**, each on a new line, separated by **"|"**, until the **"Stop"** command is given:

- **"Add|{piece}|{composer}|{key}"**:
  - You need to **add the given piece** with the information about it to the other pieces and print:
    "{piece} by {composer} in {key} added to the collection!"
  - If the piece **is already in the collection**, print:
    "{piece} is already in the collection!"
- **"Remove|{piece}"**:
  - If the piece is in the collection, **remove it** and print:
    "Successfully removed {piece}!"
  - Otherwise, print:
    "Invalid operation! {piece} does not exist in the collection."
- **"ChangeKey|{piece}|{new key}"**:
  - If the piece is in the collection, **change its key with the given one** and print:
    "Changed the key of {piece} to {new key}!"
  - Otherwise, print:
    "Invalid operation! {piece} does not exist in the collection."

Upon receiving the **"Stop"** command, you need to print all pieces in your collection in the following format:
"{Piece} -> Composer: {composer}, Key: {key}"

Input/Constraints

- You will receive **a single integer** at first – **the initial number of pieces in the collection**
- For each piece, you will receive a single line of text with information about it.
- Then you will receive multiple commands in the way described above until the command **"Stop"**.

Output

- All the output messages with the appropriate formats are described in the problem description.

Examples

Input	Output
3 Fur Elise|Beethoven|A Minor	Sonata No.2 by Chopin in B Minor added to the collection!

Moonlight Sonata|Beethoven|C# Minor Clair de Lune|Debussy|C# Minor Add|Sonata No.2|Chopin|B Minor Add|Hungarian Rhapsody No.2|Liszt|C# Minor Add|Fur Elise|Beethoven|C# Minor Remove|Clair de Lune ChangeKey|Moonlight Sonata|C# Major Stop	Hungarian Rhapsody No.2 by Liszt in C# Minor added to the collection! Fur Elise is already in the collection! Successfully removed Clair de Lune! Changed the key of Moonlight Sonata to C# Major! Fur Elise -> Composer: Beethoven, Key: A Minor Moonlight Sonata -> Composer: Beethoven, Key: C# Major Sonata No.2 -> Composer: Chopin, Key: B Minor Hungarian Rhapsody No.2 -> Composer: Liszt, Key: C# Minor

Comments

After we receive the initial pieces with their info, we start receiving commands. The first two commands are to add a piece to the collection, and since the pieces are not already added, we manage to add them. The third add command, however, **attempts to add a piece, which is already in the collection**, so we print a special message and don't add the piece. After that, we receive the remove command, and since the piece is in the collection, we remove it successfully.
Finally, the last command says to change the key of a piece. Since the key is present in the collection, we modify its key.
We receive the Stop command, print the information about the pieces, and the program ends.

Input	Output
4 Eine kleine Nachtmusik|Mozart|G Major La Campanella|Liszt|G# Minor The Marriage of Figaro|Mozart|G Major Hungarian Dance No.5|Brahms|G Minor Add|Spring|Vivaldi|E Major Remove|The Marriage of Figaro Remove|Turkish March ChangeKey|Spring|C Major Add|Nocturne|Chopin|C# Minor Stop	Spring by Vivaldi in E Major added to the collection! Successfully removed The Marriage of Figaro! Invalid operation! Turkish March does not exist in the collection. Changed the key of Spring to C Major! Nocturne by Chopin in C# Minor added to the collection! Eine kleine Nachtmusik -> Composer: Mozart, Key: G Major La Campanella -> Composer: Liszt, Key: G#

	`Minor`
	`Hungarian Dance No.5 -> Composer: Brahms, Key: G Minor`
	`Spring -> Composer: Vivaldi, Key: C Major`
	`Nocturne -> Composer: Chopin, Key: C# Minor`

JS Examples

Input	Output

`[`
`  '3',`
`  'Fur Elise|Beethoven|A Minor',`
`  'Moonlight Sonata|Beethoven|C# Minor',`
`  'Clair de Lune|Debussy|C# Minor',`
`  'Add|Sonata No.2|Chopin|B Minor',`
`  'Add|Hungarian Rhapsody No.2|Liszt|C# Minor',`
`  'Add|Fur Elise|Beethoven|C# Minor',`
`  'Remove|Clair de Lune',`
`  'ChangeKey|Moonlight Sonata|C# Major',`
`  'Stop'`
`]`
	`Sonata No.2 by Chopin in B Minor added to the collection!`
	`Hungarian Rhapsody No.2 by Liszt in C# Minor added to the collection!`
	`Fur Elise is already in the collection!`
	`Successfully removed Clair de Lune!`
	`Changed the key of Moonlight Sonata to C# Major!`
	`Fur Elise -> Composer: Beethoven, Key: A Minor`
	`Moonlight Sonata -> Composer: Beethoven, Key: C# Major`
	`Sonata No.2 -> Composer: Chopin, Key: B Minor`
	`Hungarian Rhapsody No.2 -> Composer: Liszt, Key: C# Minor`

Comments	
After we receive the initial pieces with their info, we start receiving commands. The first two commands are to add a piece to the collection, and since the pieces are not already added, we manage to add them. The third add command, however, **attempts to add a piece, which is already in the collection**, so we print a special message and don't add the piece. After that, we receive the remove command, and since the piece is in the collection, we remove it successfully. Finally, the last command says to change the key of a piece. Since the key is present in the collection, we modify its key. We receive the Stop command, print the information about the pieces, and the program ends.	

Input	Output

`[`
`  '4',`
`  'Eine kleine Nachtmusik|Mozart|G Major',`
`  'La Campanella|Liszt|G# Minor',`
`  'The Marriage of Figaro|Mozart|G Major',`
`  'Hungarian Dance No.5|Brahms|G Minor',`
`  'Add|Spring|Vivaldi|E Major',`
`  'Remove|The Marriage of Figaro',`
`  'Remove|Turkish March',`
`  'ChangeKey|Spring|C Major',`
`  'Add|Nocturne|Chopin|C# Minor',`
`  'Stop'`
`]`
	`Spring by Vivaldi in E Major added to the collection!`
	`Successfully removed The Marriage of Figaro!`
	`Invalid operation! Turkish March does not exist in the collection.`
	`Changed the key of Spring to C Major!`
	`Nocturne by Chopin in C# Minor added to the collection!`
	`Eine kleine Nachtmusik -> Composer: Mozart, Key: G Major`
	`La Campanella -> Composer: Liszt, Key: G# Minor`
	`Hungarian Dance No.5 -> Composer: Brahms, Key: G Minor`
	`Spring -> Composer: Vivaldi, Key: C Major`
	`Nocturne -> Composer: Chopin, Key: C# Minor`