

Programming Fundamentals Final Exam Preparation 1

Problem 1. World Tour

Link: <https://judge.softuni.org/Contests/Practice/Index/2518#0>

You are a world traveler, and your next goal is to make a world tour. To do that, you have to plan out everything first. To start with, you would like to plan out all of your stops where you will have a break.

On the **first line**, you will be given a string containing all of your **stops**. Until you receive the command **"Travel"**, you will be given some commands to **manipulate** that initial string. The **commands can be**:

- **"Add Stop:{index}:{string}"**:
 - Insert the given **string** at that **index** only if the index is **valid**
- **"Remove Stop:{start_index}:{end_index}"**:
 - Remove the elements of the string from the **starting index** to the **end index (inclusive)** if **both** indices are **valid**
- **"Switch:{old_string}:{new_string}"**:
 - If the **old string** is in the initial string, **replace** it with the **new one** (all occurrences)

Note: After each command, print the current state of the string

After the **"Travel"** command, print the following: **"Ready for world tour! Planned stops: {string}"**

Input / Constraints

- **JavaScript**: you will receive a **list of strings**
- An **index is valid** if it is **between the first and the last element index (inclusive)** in the sequence.

Output

- Print the proper output messages in the proper cases as described in the problem description

Examples

Input	Output
(["Hawai::Cyprys-Greece", "Add Stop:7:Rome", "Remove Stop:11:16", "Switch:Hawai:Bulgaria", "Travel"])	Hawai::RomeCyprys-Greece Hawai::Rome-Greece Bulgaria::Rome-Greece Ready for world tour! Planned stops: Bulgaria::Rome-Greece

Problem 2 – Fancy Barcode

Link: <https://judge.softuni.org/Contests/Practice/Index/2303#1>

Your first task is to determine if the given sequence of characters is a **valid** barcode or **not**.

Each line must not contain anything else but a valid barcode. A barcode is **valid** when:

- It is surrounded by a "@" followed by one or more "#"
- It is **at least 6 characters long** (without the surrounding "@" or "#")
- **It starts** with a **capital letter**
- It contains **only letters** (lower and upper case) **and digits**
- **It ends** with a **capital letter**

Examples of valid barcodes: @###Che46sE@##, @#FreshFisH@#, @###Brea0D@###, @##Che46sE@##

Examples of invalid barcodes: ##InvalidItem##, @InvalidItem@, @#Invalid_Item@#

Next, you have to determine the **product group** of the item from the **barcode**. The product group is obtained by **concatenating all the digits** found in the barcode. If there are **no digits** present in the barcode, the **default** product group is "00".

Examples:

@#FreshFisH@# -> product group: 00

@###Brea0D@### -> product group: 0

@##Che4s6E@## -> product group: 46

Input

On the first line, you will be given an integer **n** – the count of barcodes that you will be receiving next.

On the following **n** lines, you will receive different strings.

Output

For each barcode that you process, you need to print a message.

If the barcode is invalid:

- "Invalid barcode"

If the barcode is valid:

- "Product group: {product group}"

Examples

Input	Output
(["3", "@#FreshFisH@#", "@###Brea0D@###", "@##Che4s6E@##"])	Product group: 00 Product group: 0 Product group: 46
Input	Output

("6",	Product group: 11
"@###ValidItem@###",	Product group: 00
"@#ValidItem@#",	Invalid barcode
"##InvalidItem##",	Invalid barcode
"@InvalidItem@",	Invalid barcode
"@#Invalid_Item@#",	Product group: 00
"@#ValidItem@#"])	

Problem 3 – The Pianist

Link: <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', '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	
<p>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.</p> <p>Finally, the last command says to change the key of a piece. Since the key is present in the collection, we modify its key.</p> <p>We receive the Stop command, print the information about the pieces, and the program ends.</p>	
Input	Output
['4',	Spring by Vivaldi in E Major added to the collection!

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