Final Exam Preparation – 19 July 2023

1. Activation Keys

Link: https://judge.softuni.bg/Contests/Practice/Index/2302#0

You are about to make some good money, but first you need to think of a way to verify who paid for your product and who didn`t. You have decided to let people use the software for a free trial period and then require an activation key in order to continue to use the product. The last step before you could cash out is to design a program that creates unique activation keys for each user. So, waste no more time and start typing!

The first line of the input will be your raw activation key. It will consist of **letters and numbers only**.

After that, until the **"Generate"** command is given, you will be receiving strings with instructions for different operations that need to be performed upon the raw activation key.

There are several types of instructions, split by **">>>"**:

- **Contains>>>{substring}** – checks if the raw activation key contains the given substring.
  - If it does prints: **"{raw activation key} contains {substring}"**.
  - If not, prints: **"Substring not found!"**
- **Flip>>>Upper/Lower>>>{startIndex}>>>{endIndex}**
  - Changes the substring **between the given indices (the end index is exclusive)** to upper or lower case.
  - All given indexes will be valid.
  - Prints the activation key.
- **Slice>>>{startIndex}>>>{endIndex}**
  - **Deletes** the characters between the start and end indices (**end index is exclusive**).
  - Both indices will be **valid**.
  - Prints the activation key.

Input

- The first line of the input will be string and it will consist of **letters and numbers only**.
- After the first line, until the **"Generate"** command is given, you will be receiving **strings**.

Output

- After the **"Generate"** command is received, print:
  - **"Your activation key is: {activation key}"**

Examples

Input	Output

abcdefghijklmnopqrstuvwxyz Slice>>>2>>>6 Flip>>>Upper>>>3>>>14 Flip>>>Lower>>>5>>>7 Contains>>>def Contains>>>deF Generate	abghijklmnopqrstuvwxyz abgHIJKLMNOPQRstuvwxyz abgHIjkLMNOPQRstuvwxyz Substring not found! Substring not found! Your activation key is: abgHIjkLMNOPQRstuvwxyz

Comments

1. **Slice>>2>>6**

abcdefghijklmnopqrstuvwxyz becomes abghijklmnopqrstuvwxyz

2. **Flip>>>Upper>>>3>>>14**

abghijklmnopqrstuvwxyz becomes abgHIJKLMNOPQRstuvwxyz

3. **Flip>>>Lower>>>5>>>7**

abgHIJKLMNOPQRstuvwxyz becomes abgHIjkLMNOPQRstuvwxyz

4. **Contains>>>def**

abgHIjkLMNOPQRstuvwxyz does not contain def

5. **Contains>>>deF**

abgHIjkLMNOPQRstuvwxyz does not contain deF

The final activation key is abgHIjkLMNOPQRstuvwxyz

Input	Output
134softsf5ftuni2020rockz42 Slice>>>3>>>7 Contains>>>-rock Contains>>>-uni- Contains>>>-rocks Flip>>>Upper>>>2>>>8 Flip>>>Lower>>>5>>>11 Generate	134sf5ftuni2020rockz42 Substring not found! Substring not found! Substring not found! 134SF5FTuni2020rockz42 134SF5ftuni2020rockz42 Your activation key is: 134SF5ftuni2020rockz42

2. Fancy Barcodes

Link: https://judge.softuni.bg/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:

- Is surrounded with a **"@"** followed by one or more **"#"**
- Is **at least 6 characters long** (without the surrounding **"@"** or **"#"**)
- **Starts** with a **capital letter**
- Contains **only letters** (lower and upper case) **and digits**
- **Ends** with a **capital letter**

Examples of valid barcodes: **@#FreshFisH@#**, **@###Brea0D@###**, **@##Che46sE@##**, **@##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 next **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 @###Val1d1teM@### @#ValidIteM@# ##InvaliDiteM## @InvalidIteM@ @#Invalid_IteM@# @#ValiditeM@#	Product group: 11 Product group: 00 Invalid barcode Invalid barcode Invalid barcode Product group: 00

3. Plant Discovery

Link: https://judge.softuni.bg/Contests/Practice/Index/2518#2

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 of the command 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_name}; Rarity: {rarity}; Rating: {average_rating formatted to the 2nd digit}
…
```

The plants should be **sorted** by **rarity descending**, then by **average rating descending**

Input / Constraints

- You will recive 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 Update: Woodii - 5 Reset: Arnoldii Exhibition	Plants for the exhibition: - Woodii; Rarity: 5; Rating: 7.50 - Arnoldii; Rarity: 4; Rating: 0.00 - Welwitschia; Rarity: 2; Rating: 7.00
2 Candelabra<->10 Oahu<->10 Rate: Oahu - 7 Rate: Candelabra - 6 Exhibition	Plants for the exhibition: - Oahu; Rarity: 10; Rating: 7.00 - Candelabra; Rarity: 10; Rating: 6.00