# Final Exam Preparation – 23 November 2022

## Activation Keys

**Link:** [**https://judge.softuni.org/Contests/Practice/Index/2302#0**](https://judge.softuni.org/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 to continue using the product. Before you can cash out, the last step 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}":
  + If the raw activation key contains the given substring, prints: "{raw activation key} contains {substring}".
  + Otherwise, 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 and then prints the activation key.
  + All given indexes will be valid.
* **"Slice>>>{startIndex}>>>{endIndex}**":
  + **Deletes** the characters between the start and end indices (**the end index is exclusive) and** prints the activation key.
  + Both indices will be **valid**.

### Input

* The first line of the input will be a string consisting 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   1. **Flip>>>Upper>>>3>>>14**   abghijklmnopqrstuvwxyz becomes abgHIJKLMNOPQRstuvwxyz   1. **Flip>>>Lower>>>5>>>7**   abgHIJKLMNOPQRstuvwxyz becomes abgHIjkLMNOPQRstuvwxyz   1. **Contains>>>def**   abgHIjkLMNOPQRstuvwxyz does not contain def   1. **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 |

## Fancy Barcodes

**Link:** [**https://judge.softuni.org/Contests/Practice/Index/2303#1**](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  @###Val1d1teM@###  @#ValidIteM@#  ##InvaliDiteM##  @InvalidIteM@  @#Invalid\_IteM@#  @#ValiditeM@# | Product group: 11  Product group: 00  Invalid barcode  Invalid barcode  Invalid barcode  Product group: 00 |

## Plant Discovery

**Link:** [**https://judge.softuni.org/Contests/Practice/Index/2518#2**](https://judge.softuni.org/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 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.
* JavaScript: you will receive a list of strings.

### 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:  - Arnoldii; Rarity: 4; Rating: 0.00  - Woodii; Rarity: 5; Rating: 7.50  - Welwitschia; Rarity: 2; Rating: 7.00 |
| 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 |