# Final Exam Preparation – 21 November 2024

## The Imitation Game

**Link:** [https://alpha.judge.softuni.org/contests/01-programming-fundamentals-final-exam-retake/2525/practice#0](https://alpha.judge.softuni.org/contests/01-programming-fundamentals-final-exam-retake/2525/practice" \l "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  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? |

## Fancy Barcodes

**Link:** [https://alpha.judge.softuni.org/contests/04-programming-fundamentals-final-exam/2303/practice#20](https://alpha.judge.softuni.org/contests/04-programming-fundamentals-final-exam/2303/practice" \l "20)

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 |
| 6  @###Val1d1teM@###  @#ValidIteM@#  ##InvaliDiteM##  @InvalidIteM@  @#Invalid\_IteM@#  @#ValiditeM@# | Product group: 11  Product group: 00  Invalid barcode  Invalid barcode  Invalid barcode  Product group: 00 |

## The Pianist

**Link:** [https://alpha.judge.softuni.org/contests/01-programming-fundamentals-final-exam-retake/2525/practice#2](https://alpha.judge.softuni.org/contests/01-programming-fundamentals-final-exam-retake/2525/practice" \l "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  Hungarian Rhapsody No.2 -> Composer: Liszt, Key: C# Minor  Moonlight Sonata -> Composer: Beethoven, Key: C# Major  Sonata No.2 -> Composer: Chopin, Key: B 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, sorted in the way described above, 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  Hungarian Dance No.5 -> Composer: Brahms, Key: G Minor  La Campanella -> Composer: Liszt, Key: G# Minor  Nocturne -> Composer: Chopin, Key: C# Minor  Spring -> Composer: Vivaldi, Key: C Major |