# Final Exam Preparation – 22 November 2023

## World Tour

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

*You are a world traveller and your next goal is to make a world tour. In order 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}"**

### 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 |
| Albania:Bulgaria:Cyprus:Deuchland  Add Stop:3:Nigeria  Remove Stop:4:8  Switch:Albania: Azərbaycan  Travel | AlbNigeriaania:Bulgaria:Cyprus:Deuchland  AlbNaania:Bulgaria:Cyprus:Deuchland  AlbNaania:Bulgaria:Cyprus:Deuchland  Ready for world tour! Planned stops: AlbNaania:Bulgaria:Cyprus:Deuchland |

## Emoji Detector

**Link:** [**https://judge.softuni.bg/Contests/Practice/Index/2302#1**](https://judge.softuni.bg/Contests/Practice/Index/2302#1)

Your task is to write program which extracts emojis from a text and find the threshold based on the input.

You have to get your **cool threshold**. It is obtained by **multiplying all** the digits found in the input. The cool threshold could be a **very big number**, so be mindful.

An emoji is valid when:

* Is surrounded by either :: or \*\* (exactly 2)
* Is **at least 3** characters long (**without** the surrounding symbols)
* **Starts** with a **capital letter**
* Continues with **lowercase** letters **only**

Examples of valid emojis**:** ::Joy::, \*\*Banana\*\*, ::Wink::

Examples of invalid emojis: ::Joy\*\*, **::fox:es:**, **\*\*Monk3ys\*\*, :Snak::Es::**

You need to count **all valid emojis** in the text and calculate their **coolness**. The coolness of the emoji is **determined** by summing all the **ASCII values of all letters** in the emoji.

Examples: ::Joy:: - 306, \*\*Banana\*\* - 577, ::Wink:: - 409

You need to print the result of cool threshold and after that to take all emojis out of the text, count them and print the **only the cool ones** on the console.

### Input

* On the single input you will receive a piece of string.

### Output

* On the first line of the output print the obtained Cool threshold in format:
* **Cool threshold: {coolThresholdSum}**

On the next line **print the** **count of all emojis** found in the text in format:

* {countOfAllEmojis} emojis found in the text. The cool ones are:
* {cool emoji 1}
* {cool emoji 2}
* {…}

If there are no cool ones, just don't print anything in the end.

### Constraints

There will always be at least one digit in the text!

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| In the Sofia Zoo there are 311 animals in total! ::Smiley:: This includes 3 \*\*Tigers\*\*, 1 ::Elephant:, 12 \*\*Monk3ys\*\*, a \*\*Gorilla::, 5 ::fox:es: and 21 different types of :Snak::Es::. ::Mooning:: \*\*Shy\*\* | Cool threshold: 540  4 emojis found in the text. The cool ones are:  ::Smiley::  \*\*Tigers\*\*  ::Mooning:: |
| **Comments** | |
| You can see all the valid emojis in green. There are various reasons why the rest are not valid, examine them carefully. The "cool threshold" is 3\*1\*1\*3\*1\*1\*2\*3\*5\*2\*1 = 540.  ::Smiley:: -> 83 + 109 + 105 + 108 + 101 + 121 = 627 > 540 -> cool  \*\*Tigers\*\* -> 84 + 105 + 103 + 101 + 114 + 115 = 622 > 540 -> cool  ::Mooning:: -> 77 + 111 + 111 + 110 + 105 + 110 + 103 = 727 > 540 -> cool  \*\*Shy\*\* -> 83 + 104 + 121 = 308 < 540 -> not cool  At the end we print the count of all valid emojis found and each of the cool ones on a new line. | |
| **Input** | **Output** |
| 5, 4, 3, 2, 1, go! The 1-th consecutive banana-eating contest has begun! ::Joy:: \*\*Banana\*\* ::Wink:: \*\*Vali\*\* ::valid\_emoji:: | Cool threshold: 120  4 emojis found in the text. The cool ones are:  ::Joy::  \*\*Banana\*\*  ::Wink::  \*\*Vali\*\* |
| **Input** | **Output** |
| It is a long established fact that 1 a reader will be distracted by 9 the readable content of a page when looking at its layout. The point of using ::LoremIpsum:: is that it has a more-or-less normal 3 distribution of 8 letters, as opposed to using 'Content here, content 99 here', making it look like readable \*\*English\*\*. | Cool threshold: 17496  1 emojis found in the text. The cool ones are: |
| **Comments** | |
| You can see \*\*English\*\* is a valid emoji, but the sum of ascii **is not** **bigger** than cool threshold, that's why we **don't** print anything in the end. | |

## The Pianist

**Link:** [**https://judge.softuni.org/Contests/Practice/Index/2525#2**](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  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 |

*Suggestion: Choose a piece from the ones here to listen to while you are doing the problem!*