# Final Exam Preparation – 22 March 2023

## The Imitation Game

**Link:** [**https://judge.softuni.org/Contests/Practice/Index/2525#0**](https://judge.softuni.org/Contests/Practice/Index/2525#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? |

## Emoji Detector

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

Your task is to write a program that 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 **huge number**, so be mindful.

An emoji is valid when:

* It is surrounded by 2 characters, either "::" or "\*\*"
* It is **at least 3** characters long (**without** the surrounding symbols)
* **It 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 the cool threshold and, after that to take all emojis out of the text, count them and print **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 the format:

**"Cool threshold: {coolThresholdSum}"**

* On the following 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}

…

{cool emoji N}"

### 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  In 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 the cool threshold. That's why we **don't** print anything in the end. | |

## Need for Speed III

**Link:** [**https://judge.softuni.org/Contests/Practice/Index/2307#2**](https://judge.softuni.org/Contests/Practice/Index/2307#2)

*You have just bought the latest and greatest computer game – Need for Seed III. Pick your favorite cars and drive them all you want! We know that you can't wait to start playing.*

On the first line of the standard input, you will receive an integer **n** – the **number of cars** that you can obtain. On the next **n** lines, the **cars themselves** will follow with their **mileage** and **fuel** **available**, separated by "|" in the following format:

"{car}|{mileage}|{fuel}"

Then, you will be receiving different **commands**, each on a new line, separated by " : ", until the "Stop" command is given:

* "Drive : {car} : {distance} : {fuel}**"**:
  + You need to **drive the given distance**, and you will **need the given** fuel to do that. If the car **doesn't have enough fuel**, print: "**Not enough fuel to make that ride**"
  + If the car has the required fuel available in the tank, **increase its mileage** with **the given distance**, **decrease its fuel with the given fuel,** and **print**:   
    "{car} driven for {distance} kilometers. {fuel} liters of fuel consumed."
  + You like driving new cars only, so if a car's mileage reaches **100 000** km, remove it from the collection(s) and print: "**Time to sell the {car}!**"
* "Refuel : {car} : {fuel}**"**:
  + **Refill** the tank of your car.
  + Each tank can hold a **maximum of 75 liters of fuel**, so if the given amount of fuel is more than you can fit in the tank, take only what is required to fill it up.
  + Print a message in the following format: "{car} refueled with {fuel} liters"
* "Revert : {car} : {kilometers}**"**:
  + Decrease the **mileage** of the given **car with the given kilometers** and print the kilometers you have decreased it with in the following format:  
    "{car} mileage decreased by {amount reverted} kilometers"
  + If the mileage becomes **less** **than** **10 000km** **after** it is decreased, **just set it to 10 000km** and   
    **DO NOT print anything.**

Upon receiving the "Stop" command, you need to print all cars in your possession in the following format:  
"**{car} -> Mileage: {mileage} kms, Fuel in the tank: {fuel} lt.**"

### Input/Constraints

* The **mileage** and **fuel** of the cars will be valid, 32-bit integers, and will never be negative.
* The **fuel** and **distance** amounts **in the commands will never be negative**.
* The **car** **names** in the **commands** will always be **valid cars in your possession**.

### Output

* All the output messages with the appropriate formats are described in the problem description.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  Audi A6|38000|62  Mercedes CLS|11000|35  Volkswagen Passat CC|45678|5  Drive : Audi A6 : 543 : 47  Drive : Mercedes CLS : 94 : 11  Drive : Volkswagen Passat CC : 69 : 8  Refuel : Audi A6 : 50  Revert : Mercedes CLS : 500  Revert : Audi A6 : 30000  Stop | Audi A6 driven for 543 kilometers. 47 liters of fuel consumed.  Mercedes CLS driven for 94 kilometers. 11 liters of fuel consumed.  Not enough fuel to make that ride  Audi A6 refueled with 50 liters  Mercedes CLS mileage decreased by 500 kilometers  Audi A6 -> Mileage: 10000 kms, Fuel in the tank: 65 lt.  Mercedes CLS -> Mileage: 10594 kms, Fuel in the tank: 24 lt.  Volkswagen Passat CC -> Mileage: 45678 kms, Fuel in the tank: 5 lt. |
| **Comments** | |
| After we receive the cars with their mileage and fuel, we start driving them. When we get to "**Drive : Volkswagen Passat CC : 69 : 8**" command, our program calculates that there is not enough fuel, and we print the appropriate message. Then we refuel the Audi A6 with 50 l of fuel and Revert the Mercedes with 500 kilometers.  When we receive the "Revert : Audi A6 : 30000", we set its mileage to **10000** km, because if the current mileage of the Audi is **38543** kms and if we subtract **30000** from it, we receive **8543** kms, which is less than 10000 kms.  After all the commands, we print our current collection of cars with their current mileage and current fuel. | |
| **Input** | **Output** |
| 4  Lamborghini Veneno|11111|74  Bugatti Veyron|12345|67  Koenigsegg CCXR|67890|12  Aston Martin Valkryie|99900|50  Drive : Koenigsegg CCXR : 382 : 82  Drive : Aston Martin Valkryie : 99 : 23  Drive : Aston Martin Valkryie : 2 : 1  Refuel : Lamborghini Veneno : 40  Revert : Bugatti Veyron : 2000  Stop | Not enough fuel to make that ride  Aston Martin Valkryie driven for 99 kilometers. 23 liters of fuel consumed.  Aston Martin Valkryie driven for 2 kilometers. 1 liters of fuel consumed.  Time to sell the Aston Martin Valkryie!  Lamborghini Veneno refueled with 1 liters  Bugatti Veyron mileage decreased by 2000 kilometers  Lamborghini Veneno -> Mileage: 11111 kms, Fuel in the tank: 75 lt.  Bugatti Veyron -> Mileage: 10345 kms, Fuel in the tank: 67 lt.  Koenigsegg CCXR -> Mileage: 67890 kms, Fuel in the tank: 12 lt. |