Problem 1 - Activation Keys

Problem for exam preparation for the Programming Fundamentals Course @SoftUni.
Submit your solutions in the SoftUni judge system at 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

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

JS 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`

© SoftUni – about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.

Follow us:

Problem 2 - Emoji Detector

Problem for exam preparation for the Programming Fundamentals Course @SoftUni.
Submit your solutions in the SoftUni judge system at 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

© SoftUni – about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.
Follow us:

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.

JS 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`	`Cool threshold: 17496`
	`1 emojis found in the text. The cool ones are:`

`here', making it look like readable **English**."])`	

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.

© SoftUni – about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.

Follow us:

Problem 3 - P!rates

Problem for exam preparation for the Programming Fundamentals Course @SoftUni.
Submit your solutions in the SoftUni judge system at https://judge.softuni.org/Contests/Practice/Index/2302#2.

Anno 1681. The Caribbean. The golden age of piracy. You are a well-known pirate captain by the name of Jack Daniels. Together with your comrades Jim (Beam) and Johnny (Walker), you have been roaming the seas, looking for gold and treasure... and the occasional killing, of course. Go ahead, target some wealthy settlements and show them the pirate's way!

Until the **"Sail"** command is given, you will be receiving:

- You and your crew have targeted **cities**, with their **population** and **gold**, separated by **"||"**.
- If you receive a city that has already been received, you have to increase the population and gold with the given values.

After the **"Sail"** command, you will start receiving lines of text representing events until the **"End"** command is given.

Events will be in the following format:

- **"Plunder=>{town}=>{people}=>{gold}"**
  - You have successfully attacked and plundered the town, killing the given number of people and stealing the respective amount of gold.
  - For every town you attack print this message: **"{town} plundered! {gold} gold stolen, {people} citizens killed."**
  - If any of those two values (population or gold) **reaches zero**, the town is disbanded.
    - You need to **remove it** from your collection of targeted cities and print the following message: **"{town} has been wiped off the map!"**
  - There will be no case of receiving more people or gold than there is in the city.
- **"Prosper=>{town}=>{gold}"**
  - There has been dramatic economic growth in the given city**, increasing its treasury** by the given amount of gold.
  - The gold amount **can be a negative number, so be careful.** If a negative amount of gold is given, print: **"Gold added cannot be a negative number!" and ignore the command.**
  - If the given gold is a valid amount, increase the town's gold reserves by the respective amount and print the following message:

"{gold added} gold added to the city treasury. {town} now has {total gold} gold."

Input

- On the first lines, until the **"Sail"** command, you will be receiving strings representing the cities with their gold and population, separated by **"||"**
- On the following lines, until the **"End"** command, you will be receiving strings representing the actions described above, separated by **"=>"**

© SoftUni – about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.
Follow us:

Output

- After receiving the **"End"** command, if there are any existing settlements on your list of targets, you need to print all of them, in the following format:

```
"Ahoy, Captain! There are {count} wealthy settlements to go to:
{town1} -> Population: {people} citizens, Gold: {gold} kg
{town2} -> Population: {people} citizens, Gold: {gold} kg
…
{town…n} -> Population: {people} citizens, Gold: {gold} kg"
```

- If there are no settlements left to plunder, print:

```
"Ahoy, Captain! All targets have been plundered and destroyed!"
```

Constraints

- The initial population and gold of the settlements will be valid 32-bit integers, never negative, or exceed the respective limits.
- The town names in the events will always be valid towns that should be on your list.

Examples

Input	Output
Tortuga||345000||1250 Santo Domingo||240000||630 Havana||410000||1100 Sail Plunder=>Tortuga=>75000=>380 Prosper=>Santo Domingo=>180 End	Tortuga plundered! 380 gold stolen, 75000 citizens killed. 180 gold added to the city treasury. Santo Domingo now has 810 gold. Ahoy, Captain! There are 3 wealthy settlements to go to: Tortuga -> Population: 270000 citizens, Gold: 870 kg Santo Domingo -> Population: 240000 citizens, Gold: 810 kg Havana -> Population: 410000 citizens, Gold: 1100 kg
Input	**Output**

© SoftUni – about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.

Follow us:

Nassau||95000||1000 San Juan||930000||1250 Campeche||270000||690 Port Royal||320000||1000 Port Royal||100000||2000 Sail Prosper=>Port Royal=>-200 Plunder=>Nassau=>94000=>750 Plunder=>Nassau=>1000=>150 Plunder=>Campeche=>150000=>690 End	Gold added cannot be a negative number! Nassau plundered! 750 gold stolen, 94000 citizens killed. Nassau plundered! 150 gold stolen, 1000 citizens killed. Nassau has been wiped off the map! Campeche plundered! 690 gold stolen, 150000 citizens killed. Campeche has been wiped off the map! Ahoy, Captain! There are 2 wealthy settlements to go to: San Juan -> Population: 930000 citizens, Gold: 1250 kg Port Royal -> Population: 420000 citizens, Gold: 3000 kg

JS Examples

Input	Output
(["Tortuga||345000||1250", "Santo Domingo||240000||630", "Havana||410000||1100", "Sail", "Plunder=>Tortuga=>75000=>380", "Prosper=>Santo Domingo=>180", "End"])	Tortuga plundered! 380 gold stolen, 75000 citizens killed. 180 gold added to the city treasury. Santo Domingo now has 810 gold. Ahoy, Captain! There are 3 wealthy settlements to go to: Tortuga -> Population: 270000 citizens, Gold: 870 kg Santo Domingo -> Population: 240000 citizens, Gold: 810 kg Havana -> Population: 410000 citizens, Gold: 1100 kg
Input	**Output**

© SoftUni – about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.

Follow us:

(["Nassau||95000||1000", "San Juan||930000||1250", "Campeche||270000||690", "Port Royal||320000||1000", "Port Royal||100000||2000", "Sail", "Prosper=>Port Royal=>-200", "Plunder=>Nassau=>94000=>750", "Plunder=>Nassau=>1000=>150", "Plunder=>Campeche=>150000=>690", "End"])	Gold added cannot be a negative number! Nassau plundered! 750 gold stolen, 94000 citizens killed. Nassau plundered! 150 gold stolen, 1000 citizens killed. Nassau has been wiped off the map! Campeche plundered! 690 gold stolen, 150000 citizens killed. Campeche has been wiped off the map! Ahoy, Captain! There are 2 wealthy settlements to go to: San Juan -> Population: 930000 citizens, Gold: 1250 kg Port Royal -> Population: 420000 citizens, Gold: 3000 kg

© SoftUni – about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.

Follow us: