More Exercise: Regular Expressions

Problems for exercise and homework for the "C# Fundamentals" course @ SoftUni
You can check your solutions in Judge

1. Winning Ticket

The lottery is exciting. What is not, is checking a million tickets for winnings only by hand. So, you are given the task to create a program that automatically checks if a ticket is a winner.

You are given a **collection of tickets separated by commas and spaces**. You need to check every one of them if they have a winning combination of symbols.

A valid ticket should have exactly 20 characters. The winning symbols are '**@**', '**#**', '**$**' and '^'. But for a ticket to be a winner the symbol should uninterruptedly repeat at least **6 times** in both the **tickets left half** and the **tickets right half**.

For example, a valid winning ticket should be something like this:

"**Cash$$$$$$Ca$$$$$$sh**"

The left half "**Cash$$$$$$**" contains "**$$$$$$**", which is also contained in the tickets right half "**Ca$$$$$$sh**". A winning ticket should contain symbols repeating up to 10 times in both halves, which is considered a Jackpot (for example "**$$$$$$$$$$$$$$$$$$$$**").

Input

The input will be read from the console. The input consists of a **single line,** containing all tickets **separated by commas and one or more white spaces** in the format:

- **"{ticket}, {ticket}, … {ticket}"**

Output

Print the result for every ticket in the order of their appearance, each on a separate line in the format:

- **Invalid ticket** - **"invalid ticket"**
- **No match** - **"ticket "{ticket}" - no match"**
- **Match with length 6 to 9** - **"ticket "{ticket}" - {match length}{match symbol}"**
- **Match with length 10** - **"ticket "{ticket}" - {match length}{match symbol} Jackpot!"**

Constrains

- The number of tickets will be in the range [0…100].

Examples

Input	Output
Cash$$$$$$Ca$$$$$$sh	ticket "Cash$$$$$$Ca$$$$$$sh" - 6$
$$$$$$$$$$$$$$$$$$$$, aabb , th@@@@@@eemo@@@@@@ey	ticket "$$$$$$$$$$$$$$$$$$$$" - 10$ Jackpot! invalid ticket ticket "th@@@@@@eemo@@@@@@ey" - 6@
validticketnomatch:(	ticket "validticketnomatch:(" - no match

2. Rage Quit

Every gamer knows what rage-quitting means. It's basically when you're just not good enough and you blame everybody else for losing a game. You press the CAPS LOCK key on the keyboard and flood the chat with gibberish to show your frustration.

Chochko is a gamer and a bad one at that. He asks for your help – he wants to be the most annoying kid on his team, so when he rage-quits he wants something truly spectacular. He'll give you **a series of strings followed by non-negative numbers**, e.g. "a3"; you need to print on the console **each string repeated N times**; **convert the letters to uppercase beforehand**. In the example, you need to write back "AAA".

On the output, print first a statistic of the **number of unique symbols** used (the casing of letters is irrelevant, meaning that '**a**' and '**A**' are the same); the format should be **"`Unique symbols used {0}`"**. Then, **print the rage message** itself.

The **strings and numbers will not be separated by anything**. The input will always start with a string and for each string, there will be a corresponding number. The entire input will be given on a **single line**; Chochko is too lazy to make your job easier.

Input

- The input data should be read from the console.
- It consists of a single line holding a series of **string-number sequences**.
- The input data will always be valid and in the format described. There is no need to check it explicitly.

Output

- The output should be printed on the console. It should consist of **exactly two lines**.
- On the first line, print the **number of unique symbols used** in the message.
- On the second line, print the **resulting rage message** itself.

Constraints

- The count of **string-number pairs** will be in the range [1…20000].
- Each string will contain any character **except digits**. The **length** of each string will be in the range [1…20].
- The **repeat count** for each string will be an integer in the range [0…20].
- Allowed working time for your program: 0.3 seconds. Allowed memory: 64 MB.

Examples

Input	Output	Comments
a3	`Unique symbols used: 1` AAA	We have just one string-number pair. The symbol is 'a', convert it to uppercase and repeat 3 times: AAA. Only one symbol is used ('A').
aSd2&5s@1	`Unique symbols used: 5` ASDASD&&&&&S@	"aSd" is converted to "ASD" and repeated twice; "&" is repeated 5 times; "s@" is converted to "S@" and repeated once. 5 symbols are used: 'A', 'S', 'D', '&' and '@'.

3. Post Office

You read a single line of ASCII symbols and the message is somewhere inside it, you must find it.

The input consists of three parts separated with **"|"** like this:

`"{firstPart}|{secondPart}|{thirdPart}"`

Each word **starts with a capital letter** and **has a fixed length**, you can find those in each different part of the input.

The **first part** carries the capital letters for each word inside the message. You need to find those capital letters **1 or more from A to Z**. The capital letters should be surrounded from both sides with any of the following symbols – **"#, $, %, *, &".** And those symbols **should match on both sides.** This means that **$AOTP$ - is a valid** pattern for the capital letters. **$AKTP% - is invalid since the symbols do not match.**

The **second part** of the data contains the **starting letter ASCII code and words length /between 1 – 20 characters/**, in the following format: **"`{asciiCode}:{length}`"**. For example, "**`67:05`**" – means that "67" - **ASCII code equal to the capital letter "C",** represents a word starting with "C" with the following 5 characters: like **"Carrot".** The **ASCII code** should be a **capital letter equal to a letter from the first part**. Word's length **should be exactly 2 digits**. Length **less than 10 will always have a padding zero**, **you don't need to check that.**

The **third part of the message** are **words separated by spaces.** Those **words have to start with the Capital letter [A…Z] equal to the ASCII code and have exactly the length for each capital letter you have found in the second part. Those words can contain any ASCII symbol without spaces.**

When you find a **valid word**, you have to **print it on a new line**.

Input / Constraints

- On the first line – the text is in form of three different parts separated by **"|". There can be any ASCII character inside the input, except '|'.**
- Input will always be valid - you don't need to check it.
- The input will always have three different parts, that will always be separated by '|'.

Output

- **Print all extracted words, each on a new line.**
- Allowed working **time** / **memory**: **100ms** / **16MB.**

Examples

Input	Output	Comment
sdsGGasAOTPWEEEdas$AOTP$|a65:1.2s65:03d79:01ds 84:02! -80:07++ABs90:1.1|adsaArmyd Gara So La Arm Armyw21 Argo O daOfa Or Ti Sar saTheww The Parahaos	Argo Or The Parahaos	The capital letters are "AOTP" Then we look for the addition length of the words for each capital letter. For A(65) -> it's 4. For O(79) -> it's 2. For T(84) -> it's 3. For P(80) -> it's 8. Then we search in the last part for the words. First, start with the letter 'A' and we find "Argo". With the letter 'O' we find "Or". With the letter 'T' we find "The" and with the letter 'P' we find "Parahaos".

Urgent"Message.TO$#POAML#|readData79:05:79:0!2reme80:03--23:11{79:05}tak{65:11ar}!77:!23--)77:05ACCSS76:05ad|Remedy Por Ostream :Istream Post sOffices Office Of Ankh-Morpork MR.LIPWIG Mister Lipwig	Post Office Ankh-Morpork Mister Lipwig	The first capital letters are "POAML" Then we look for the additional length of the words for each capital letter. P(80) -> it's 4. O(79) -> it's 6. A(65) -> it's 12. M(77) -> it's 6. L(76) -> it's 6. Then we search the last part for the words. First, start with the letter 'P' and we find "Post". With the letter 'O' we find "Office". With the letter 'A' we find "Ankh-Morpork". With the letter 'M' we find "Mister" and with the letter 'L' we find "Lipwig".

4. Santa's Secret Helper

After the successful second Christmas, Santa needs to gather information about the behavior of children to plan the presents for next Christmas. He has a secret helper, who is sending him **encrypted** information. Your task is to **decrypt it** and create a list of the good children.

You will receive an **integer**, which represents **a key,** and afterward some **messages**, which you **must decode** by **subtracting the key** from the **value** of **each character**. After the decryption, to be considered a valid match, a message should:

- Have a name, which **starts after '@'** and contains **only letters from the Latin alphabet**.
- Have a behavior type - **"G"(good) or "N"(naughty)** and must be **surrounded by "!"** (exclamation mark).

The order in the message should be the **child's name -> child's behavior.** They can be separated from the others by **any character except '@', '-', '!', ':' and '>'.**

You will be receiving messages until you are given the "**end**" command. Afterward, print the names of the children, who will receive a present, each on a new line.

Input / Constraints

- The **first line holds n** – the number which you have to subtract from the characters – **integer in the range [1…100].**
- On the next lines you will be receiving encrypted messages.

Output

Print the **names of the children**, each on a new line.

Examples

Input	Output	Comments
3 CNdwhamigyenumje$J$ CEreelh-nmguuejn$J$ CVwdq&gnmjkvng$Q$ end	Kate Bobbie	We receive three messages and to decrypt them we use the key: The first message has decryption key 3. So we subtract from each characters code 3 and we receive: @Kate^jfdvbkrjgb!G! @Bobbie*kjdrrbgk!G! @Stan#dkjghskd!N! **They are all valid** and they contain a child's name and behavior – G for good and N for naughty.

Input	Output	Comments
3 N}eideidmk$'(mnyenmCNlpamnin$J$ ddddkkkkmvkvmCFrqqru-nvevek$J$nmgievnge ppqmkkkmnolmnnCEhq/vkievk$Q$ yyegiivoguCYdohqwlqh/kguimhk$J$ end	Kim Connor Valentine	We receive four messages. They are with key 3: Kzbfabfajh!$%jkvbkj@Kim^jkfk!G! aaaahhhhjshsj@Connor*ksbsbh!G!kjdfbskdb mmnjhhhjklijkk@Ben,shfbsh!N! vvbdffsldr@Valentine,hdrfjeh!G!