# Problem 3. Caesar Logins

The new craze in the web dev world is to create crazy “unhackable” authentication schemes. As one of the top people in the hacker community, you want to get the credit for hacking the newest authentication scheme: the **Caesar Login**. It turns out that a Caesar Login is pretty simple to implement, but it’s quite difficult to read by humans (“security through obscurity” in the worst sense imaginable). As it happens, you’ve just set up a **man-in-the-middle** attack on one of the largest cryptocurrency trading sites, which happens to use this fictional login method and your exploit is quietly collecting Caesar Logins. Now, it’s time to crack the logins and run off with everyone’s money.

A standard **Caesar Login** looks like this:

* /tickticktocktockticktock/

***Legend****: :* ***username****, :* ***password***

Here’s the rules a **valid** **Caesar Login** should follow:

* It always **starts** and **ends** with the **one of two separators** (either / or \). It needs to be the same on both sides!
* Then, it has the username, repeated **twice** (a **username** can only contain **alphanumeric** characters)
* Then, it has the password, repeated **twice**. (a **password** can only contain **alphanumeric** characters)
* Finally, it has the **username** and **password** **one after the other**

In the example above, the **username** is “**tick**”, and the password is “**tock**”

The fun part is that each **username** and **password** can have **any character that’s invalid for a username/password** at their **start** and **end**.

Here’s another example:

* \^\muynu\_muynu^789%;789muynu789\

This one is a little bit different, as it has **non-alphanumeric** before/after the usernames and passwords. This is where the **“Caesar”** part in a Caesar Login comes in. The upper example has **6** **non-alphanumeric characters** ('^', '\', '\_', '^', '%', ';'). You need to **count** these characters and **subtract that count** from each character’s **ASCII** value in both **the username and password** in order to get their **actual values**. In the upper example, the username is “gosho” and the password is “123”

Every **Caesar Login** which **doesn’t follow** the format specified above needs to be **ignored**, as it is **invalid**.

## Input / Constraints

The input data should be read from the console.

* You will receive **Caesar Login** strings on the console, until you receive the command “/end/”

## Output

Print each **valid** Caesar Login in the **following format**:

* “user: {username}, pass: {password}”

## Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| /tickticktocktockticktock/  /wrong-wrong->cc&passpasswrong+pass/  \^\muynu\_muynu^789%;789muynu789\  \rgujq)rgujqrgpmcrgpmc+rgujqrgpmc\  /end/ | user: tick, pass: tock  user: gosho, pass: 123  user: pesho, pass: penka |
| **Input** | **Output** |
| /gg\_gg\_xxxxggxx\  \/FkcoqpfFkcoqpfjwpvgt4jwpvgt4Fkcoqpfjwpvgt4\\  /c<cd>dcd/  /end/ | user: Diamond, pass: hunter2  user: a, pass: b |
| **Input** | **Output** |
| \\ltxmt\_ltxmt^678%678`ltxmt678\  /lydqlydqLYR456789-LYR456789lydq++LYR456789/  \lshwury\ lshwuryLS6478/LS6478lshwuryLS6478\  /end/ | user: gosho, pass: 123  user: ivan, pass: IVO123456  user: ipetrov, pass: IP3145 |