# Problem 2 - Message Encrypter



Create a program that **checks** if **inputs** have a **valid message** and **encrypt** it. On the **first** line, you will **receive** a **number** that **indicates** how **many** **inputs** you will **receive** on the **following** lines**.**

A message is **valid** when:

* It is at the **end** of the input
* It **starts** with a **tag**, which is **surrounded** by either **"\*"** or **"@"** (but **not both** at the same time). The tag itself has to be a **minimum of 3** characters long, **starting** with an **uppercase** letter, **followed** **only** by **lowercase** letters
* There is a **colon** and a single **white space** after the tag
* Exactly 3 groups are consisting of **a letter** between **"["** and **"]"**, followed by a **pipe** (**"|"**)

**Example for a valid message:**

**"\*Request\*: [I]|[s]|[i]|"**

You must **check** if the **message** is **valid** and if it **is** - **encrypts** it. If it **isn't** - **print** the following **message**:

**"Valid message not found!"**

**Encrypting** a **message** means taking **all** **letters** and **turning** them **into** **ASCII** **numbers**. After successful encrypt, print it in the following format:

**"{tag}: {number1} {number2} {number3}"**

### Input

* On the **first** line, you will receive an integer **n** - the count of inputs.
* On the **following** **n** lines - **input** that you have to **check** if it has a **valid** **message**.

### Output

* Print all results from each input, each on a new line.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comment** |
| 3  \*Request\*: [I]|[s]|[i]|  \*Taggy@: [73]|[73]|[73]|  Should be valid @Taggy@: [v]|[a]|[l]| | Request: 73 115 105  Valid message not found!  Taggy: 118 97 108 | We have 3 input lines to check. The first one follows the rules and is valid. The second one doesn't because the tag is surrounded by both '\*' and '@'. The third one has a valid message and is at the end of the input. |
| 3  @Taggy@: [i]|[n]|[v]|[a]|[l]|[i]|[d]| this shouldn't be valid  \*tAGged\*: [i][i][i]|  Should be invalid @Taggy@: [v]|[a]|[l]|[l]|[l]| | Valid message not found!  Valid message not found!  Valid message not found! |  |

### JS Examples

The input will be provided as an array of strings.

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comment** |
| (["3",  "\*Request\*: [I]|[s]|[i]|",  "\*Taggy@: [73]|[73]|[73]|",  "Should be valid @Taggy@: [v]|[a]|[l]|"]) | Request: 73 115 105  Valid message not found!  Taggy: 118 97 108 | We have 3 input lines to check. The first one follows the rules and is valid. The second one doesn't because the tag is surrounded by both '\*' and '@'. The third one has a valid message and is at the end of the input. |
| (["3",  "@Taggy@: [i]|[n]|[v]|[a]|[l]|[i]|[d]| this shouldn't be valid",  "\*tAGged\*: [i][i][i]|",  "Should be invalid @Taggy@: [v]|[a]|[l]|[l]|[l]|"]) | Valid message not found!  Valid message not found!  Valid message not found! |  |