# Problem 3. Trainegram

It’s like telegram, but … between trains. You have been tasked to write a software that read encoded trainegram messages. Trainegram messages are used to send meta information about trains.

A **locomotive** consists of **2** **surrounding square brackets** and a “**less than” symbol** and a **dot** at the **end** (“<[].”). **Between** the **brackets**, the **locomotive** may contain **ANY symbol** except **letters** and **digits**.

**Valid** **locomotives**: “<[\*/\*].”, “<[---].”, “<[.;[].”...

**Invalid locomotives**: “[//].”, “<[\*-\*]”, “<[asd1].”

A **locomotive** might be **accompanied** by **several wagons**.A **wagon** consists of a **dot**, **2 surrounding square brackets** and **another** **dot** (“.[].”). **Between** the **brackets**, the **wagon** may contain **ONLY letters** and **digits**.

**Valid wagons**: “.[asd].”, “.[131].”, “.[as2].”, “.[].” ...

**Invalid wagons**: “[asd]”, “.[3-D].”.

On **each input line** you will **receive** a **message**. You should **check** if that message **IS** a **train**.

A **train ALWAYS** **has** a **locomotive** and **may** have **several wagons**.

A **train** **ALWAYS** **starts** at the **start** of the **message** and **ends** at its **end**.

A **train** **ALWAYS** **starts** with a **locomotive**. Example: “<[/\*\*]..[asd]..[3dx].”

The **input sequence** **ends** when you receive the command “Traincode!”. When that happens, you must print all **valid trains** you’ve **found**, **each** on a **new line**, by **order** of **input**.

### Input

* The input will come in the form of several lines containing messages.
* The input ends when you receive the command “Traincode!”.

### Output

* As output you must print **all valid trains**, **each** on a **new line**, by **order** of **input**.

### Constrains

* There will be no more than **1000 input lines**.
* The strings in the input lines may contain **any ASCII character**.
* The strings in the input lines may be **INVALID**.
* Allowed time / memory: 100ms / 16 MB.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| <[/\*\*]..[asd]..[3dx]..[]..[].  <>  Traincode! | <[/\*\*]..[asd]..[3dx]..[]..[]. |
| <[{]..[7]..[]..[]..[C2I43].  <[(\_#/}$)$]..[GO5A]..[G5]..[3W4].  <[^]..[54]..[S].  <[@].  <[)$-{,]..[PB1N]..[R757G].  <[]..[]..[10]..[223F]..[GBM4].  <[!]..[]  <[)\_]..[3N]..[TS]..[0NS58].  Traincode! | <[{]..[7]..[]..[]..[C2I43].  <[(\_#/}$)$]..[GO5A]..[G5]..[3W4].  <[^]..[54]..[S].  <[@].  <[)$-{,]..[PB1N]..[R757G].  <[]..[]..[10]..[223F]..[GBM4].  <[)\_]..[3N]..[TS]..[0NS58]. |

var input = Console.ReadLine();

while (input != "Traincode!")

{

var pattern = @"^(<\[[^A-Za-z0-9]\*\]\.)(\.\[[A-Za-z0-9]\*\]\.)\*$";

var match = Regex.Match(input, pattern);

if (match.Success)

{

Console.WriteLine(string.Join("", match));

}

input = Console.ReadLine();

}