Learning to code

Problem ID: learningtocode

In the 21st century, Hunters are expected to master programming languages, in order to effectively gather information on the Internet.

Gon is currently learning NenScript — the most popular programming language amongst Hunters.

Today, Gon is learning variables. In NenScript, a variable always stores a string of zero or more characters, each character is any of the following: lowercase and uppercase English letters, digits, space and special characters: **!@#$%^&*()-_=+**. In this problem, we call these characters *good characters*. The name of every variable only consists of lowercase English letters, and underscore (_). The length of a variable name is **between** 1 **and** 10, inclusive.

In order to use a variable, first we need to declare it. The syntax of the declaration is as below:

```
var <name> = <value>;
```

Here <name> is the name of the variable you want to declare, and <value> is an **expression** which denotes the string assigned to this variable. There are 3 types of expression in NenScript:

- **Variable name**, which means that the value equals to some previously declared variable.
- **String literal**, which explicitly states the value by putting its sequence of characters in quotes.
- **Template literal**, which allows you to create a string based on values of other variables by embedding expressions.

In a template literal, embedded expressions are calculated, then concatented with other parts to create a string. Template literals are enclosed by back-tick (`) and contain several (possibly zero) string expressions. String expressions are put inside curly braces following a dollar sign (${expression}). In other words, a template literal is an expression of the form `$S_1\${E_1}S_2\${E_2}\ldots S_n\${E_n}S_{n+1}$`, where $n \geq 0$. For every valid i, S_i is a string consisting of zero or more *good characters*, and E_i is an **expression**.

Let's take an example:

```
var a = "Gon";
var b = a;
var c = `My name is ${a}`;
```

Here, the values of a, b and c are "Gon", "Gon" and "My name is Gon", respectively. Note that quotes are for clarity only, no vairable's value contains any quotes.

Template literals can be nested, in other words, there can be a template literal inside a template literal. For example:

```
var a = "Gon";
var b = `My name ${`is ${a}`}`;
```

In this example, " `is ${a}` ", whose value is "is Gon", acts as an embedded expression of the template literal assigned to variable b. The value of b is "My name is Gon".

Your task is to read a sequence of commands in NenScript, each of them is either a variable declaration, as explained above; or a print request, which is in the following form, where <expr> is an expression:

```
print <expr>;
```

For each print request, print the value of the given expression.

Input

The input consists of several lines, each is either a variable declaration or a print request, as explained above. It is guaranteed all variables are neither declared twice, nor used before being declared. The input is terminated by a line with exactly one word "end.". The total length of all lines does not exceed 10^4.

It is also guaranteed that the values of the expressions only contain *good characters*.

Output

For each print request, print on a separate line the value of the corresponding expression. It is guaranteed that you have to print at least 1 and at most 10^4 characters. Please be aware that we use the **case sensitive** and **space change sensitive** checker.

Sample Input 1

```
var a = "Gon";
var b = a;
var c = `My name is ${a}`;
print c;
print `My name is ${b}`;
end.
```

Sample Output 1

```
My name is Gon
My name is Gon
```

Sample Input 2

```
var one = "1";
var two = "2";
var three = "3";
print `${one} + ${two} = ${three}`;
print `1${`2${three}2`}1`;
end.
```

Sample Output 2

```
1 + 2 = 3
12321
```