10614 Dreadful Vectors

Tanvir and Taufiq are very afraid of vectors now-a-days. It has become a regular incident to wake up from sleep after seeing the nightmare: Arrows with pointed head chasing the two friends along the road. It's not because they are weak in mathematics. Few weeks ago, as part of their Interfacing Lab project, they made a vector calculation circuit which took input strings from a PC via a parallel port and sent outputs back via a serial port. The input string is an expression consisting of vectors and the output is the result of the expression. The circuit was made of a lot of capacitors, resistors, diodes and inductors, and was very powerful. But they had never tested the circuit with illegal input strings before they showed their circuit to their teacher on the project submission date. The teacher gave a wrong expression as input, and something happened to the circuit: a resistor became red-hot due to overheating, an capacitor leaked some liquid subtance on that resistor, there were some chemical reaction between the heated leaked substance, the zinc-coated lead of a diode and the PCB coating, …, …, the power line got short circuited, …, …, (nobody knows what happened then) …, and finally, the circuit exploded with the whole lab…

What a costly input!

But the teacher is not satisfied. He wants to test more. He has told them to make the circuit again, and has assured them that he won't give wrong input strings again. But Tanvir and Taufiq wants to be safe. They want a computer program to predict the result of an input string before they sends the string to their 'circuit-bomb'.

Note: In this problem, a scalar is an integer. A vector consists of three scalar expressions separated by commas (‘`,`’) and surrounded by square brackets (‘`[`’ and ‘`]`’). As in vector algebra, addition (‘`+`’) and subtracton (‘`-`’) is defined when either both operands are scalars or both operands are vectors, and not allowed between a scalar and a vector. Multiplication (‘`*`’) is allowed for any ‘mixture’ of operands: when both operands are vectors, it means dot-product; otherwise it means scalar multiplication. Cross-product operation between two vectors is denoted by small `x` character (‘`x`’). ‘`+`’ and ‘`-`’ have the same precedence and usual associativity (left-to-right). ‘`*`’ and ‘`x`’ also have the same precedence and usual associativity, but their precedence is higher than that of ‘`+`’ and ‘`-`’. An expression inside parentheses (‘`(`’ and ‘`)`’) has higher precedence than an expression outside.

Input

Input consists of several input strings, one in each line. The end of input is marked with a line consisting of a single ‘`#`’ mark. Input may contain any character except ‘`#`’ and can be upto 100 characters long. All characters except space (ASCII 32), ‘`+`’, ‘`-`’, ‘`*`’, ‘`x`’, ‘`,`’, ‘`[`’, ‘`]`’, ‘`(`’, ‘`)`’ and digits are illegal. Integers in input are always non-negative, but the result may be negative. Intermediate and final values of scalars will have absolute value less than 2^{31}.

Output

For each input, print the value of the expression if there is no error. Otherwise, print the word ‘`Bang!`’. There should be no spaces in the output.

Sample Input

```
[1,2,3] + [4,5,6]
[ 1, 2, 3 ] * ( [ 4, 15, 6 ] * (2-1*1) )
```

```
[1,2,3] x [4,5,6] x 2
[ 1, 2 , 3, 4+5 ]
#
```

Sample Output

```
[5,7,9]
52
Bang!
Bang!
```