A student has been given the numbers in an equation, but the operations have all been erased. They must work out what the operations were to make the equation correct. Due to the sheer number of these equations that they have to work through, you have been approached to help.

Fortunately there are only four operations possible, addition (`+`), subtraction (`-`), multiplication (`*`) and division (`/`). The usual precedence of operations applies, i.e., multiplication and division are computed before addition and subtraction.

The equations are of the form

$$A = B_1 \circ B_2 \circ B_3 \circ \cdots \circ B_n,$$

where A and B_i are integers, and $\circ$ represents the positions where operations need to be inserted.

Division may only be used if integer division is possible without remainders and without dividing by zero (for example, $5/3$ is not allowed, but $6/3$ is).

You are guaranteed that exactly one solution exists for each equation.

Input

The input consists of an arbitrary number of equations, but not more than 100. Each record consists of a single line, with a list of numbers on it. The first number n ($2 \leq n \leq 10$) is the count of numbers on the right of the equals sign. The second number on the line represents A, and is followed by n numbers representing B_1, B_2, ..., B_n.

The end of input is signified by '`-1`' on its own line.

All input values are non-negative integers, and it is guaranteed that the product of the non-zero B_i's is at most 10^{18}. The `long` type in Java and `long long` type in C/C++ are sufficient to store such values.

Output

For each input record, output the equation that results, in the form shown in the sample output. There must be a single space between each number and any adjacent operators.

Sample Input

```
4 16 2 2 2 2
3 9 2 3 3
3 21 1 4 5
-1
```

Sample Output

```
16 = 2 * 2 * 2 * 2
9 = 2 * 3 + 3
21 = 1 + 4 * 5
```