Problem E. Exact Arithmetic

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Let X be the set of all rational numbers and all numbers of form $q\sqrt{r}$, where q is a non-zero rational number and r is an integer greater than 1. Here, r must not have a square number except for 1 as its divisor. Also, let X^* be the set of all numbers which can be expressed as a sum of one or more elements in X.

Consider the machine Y which is a stack-based calculator operating on the values in X^* and has the instructions shown below.

- "`push` n": pushes an integer specified in the operand onto the stack.
- "`add`": pops two values x_1 and x_2 from the top of the stack in this order, then pushes $(x_2 + x_1)$.
- "`sub`": pops two values x_1 and x_2 from the top of the stack in this order, then pushes $(x_2 - x_1)$.
- "`mul`": pops two values x_1 and x_2 from the top of the stack in this order, then pushes $(x_2 \cdot x_1)$.
- "`div`": pops two values x_1 and x_2 from the top of the stack in this order, then pushes $(x2/x1)$. Here, x_1 must be a non-zero value from X (not from X^*).
- "`sqrt`": pops one value x from the stack and pushes the square root of x. Here, x must be a non-negative rational number.
- "`disp`": pops one value x from the stack, and outputs the string representation of the value x to the display. The representation rules are stated later.
- "`stop`": terminates calculation. The stack must be empty when this instruction is called.

Initially, the stack is empty. A sufficient number of values must exist in the stack on execution of every instruction. In addition, due to the limitation of the machine Y, no more values can be pushed when the stack already stores as many as 256 values. Also, there exist several restrictions on values to be pushed onto the stack:

- For rational numbers, neither numerator nor denominator in the irreducible form may exceed 32 768 in its absolute value.
- For any element in X of the form $q\sqrt{r} = (a/b)\sqrt{r}$, the following must hold: $|a\sqrt{r}| \leq 32\,768$ and $|b| \leq 32\,768$.

For any element in X^*, each term in the sum must satisfy the above conditions.

The rules for the string representations of the values (on the machine Y) are as follows:

- A rational number is represented as either an integer or an irreducible fraction with a denominator greater than 1.
- A fraction is represented as "*num*`/`*den*". Here, *num* is the numerator, and *den* is the denominator. The sign character "`-`" precedes negative numbers.
- A number of the form $q\sqrt{r}$ is represented as "*rep*`*sqrt(`r`)`" except for the case with $|q| = 1$, in which the number is represented as `sqrt(`r`)` for $q = 1$ or `-sqrt(`r`)` for $q = -1$. Here, *rep* is the string representation of q.

- For a sum of two or more elements of X, string representations of all the (non-zero) elements are connected using the binary operator "+". In this case, all terms with the same rooted number are merged into a single term, and the terms must be shown in the ascending order of their root component. For the purpose of this rule, all rational numbers are regarded to accompany $\sqrt{1}$. There is exactly one space character before and after each of the binary operators "+". No space characters appear at any other place.

The followings are a few examples of valid string representations:

```
0
1
-1/10
2*sqrt(2) + 1/2*sqrt(3) + -1/2*sqrt(5)
1/2 + sqrt(10) + -sqrt(30)
```

Your task is to write a program that simulates the machine Y.

Input

The input is a sequence of no more than 3000 instructions. Each line contains a single instruction. You may assume that every instruction is called in a legal way. The instruction "`stop`" appears only once, at the end of the entire input.

Output

Output the strings which the machine Y will display. Write each string on a separate line.

Example

standard input	standard output
push 1 push 2 sqrt div push 1 push 3 sqrt div add disp push 8 sqrt push 3 push 2 sqrt mul add disp stop	1/2*sqrt(2) + 1/3*sqrt(3) 5*sqrt(2)