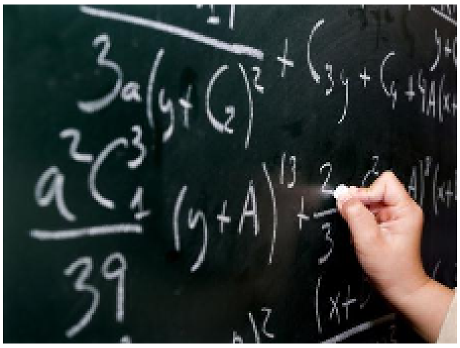
John has never been very good at maths. Due to his bad grades, his parents have sent him to the Academic Coalition of Mathematics (ACM). Despite the large amount of money his parents are spending on the ACM, John does not pay much attention during classes. However, today he has begun to think about all the effort his parents are putting into his education, and he has started to feel somewhat... guilty. So he has made a decision: he is going to improve his maths grades!



However, no sooner had he resolved to pay attention than the lesson ended. So the only thing he has been able to do is to hurriedly copy the content of the blackboard in his notebook. Today, the teacher was explaining basic arithmetic expressions with unknowns. He vaguely remembers that his classmates have been substituting values into the unknowns to obtain the expressions' results. However, in all the hurry, John has only written down expressions, values and results in a messy fashion. So he does not know which value comes with each unknown, or which result goes with each expression.

That is the reason he needs your help: he wants to know, given an expression, some values and a result, whether it is possible or not to assign those values to the unknowns in order for the expression to evaluate to the given result. The particular assignment of values does not matter to John, as he wants to do it by himself. He only wants to know whether it is possible or not.

Input

Each test case in the input file consists of two lines:

- The first line contains a sequence of natural numbers. The first one ($1 \leq n \leq 5$) is the number of unknowns that will occur in the expression. It is followed by a sequence of n integers $v_1 \ldots v_n$ ($0 \leq v_i \leq 50$), which are the values to be assigned to the unknowns. Finally, there is an integer m ($0 \leq m \leq 1000$) representing the desired result of the evaluation of the expression.
- The second line contains an arithmetic expression composed of lowercase letters (`a-z`), brackets (`(` and `)`) and binary operators (`+`, `-`, `*`). This expression will contain n unknowns, represented by n different lowercase letters, without repetitions. The expression will not contain any blanks and will always be syntactically correct, i.e. it is just an unknown or has the form (e_1 *op* e_2), where e_1 and e_2 are expressions and *op* is one of the three possible binary operators.

The input will finish with a dummy test case of just one line containing '`0 0`', which must not be processed.

Output

For each test case, print a single line with '`YES`' if there exists an assignment of the values $v_1 \ldots v_n$ to the unknowns such that the expression evaluates to m, and '`NO`' otherwise. Notice that each value v_i must be assigned to exactly one unknown.

Sample Input

```
3 2 3 4 14
((a+b)*c)
2 4 3 11
(a-b)
1 2 2
a
0 0
```

Sample Output

```
YES
NO
YES
```