Guess the Numbers

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

The first line of the input contains an integer t. t test cases follow.

Each test case consists of two lines.

The first line starts with an integer n, then n more numbers $v_1, \ldots, v_n$ and another number m. n is the number of unknowns that will occur in the expression, $v_1, \ldots, v_n$ are the values to be assigned to the unknowns, and m is 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 (+, −, $\star$). 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.

Output

For each test case, print a line containing "Case #i: x!" where i is its number, starting at 1, and x is either `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. Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 500$
- $1 \leq n \leq 5$
- $0 \leq v_i \leq 50$ for all $1 \leq i \leq n$
- $0 \leq m \leq 1000$
- The expression will be syntactically correct as described in the input section.

Sample Input 1

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

Sample Output 1

```
Case #1: YES
Case #2: NO
Case #3: YES
```