Do not get perplexed with the name of problem, it has nothing to do with integral calculus. We will concentrate on integration of simple algebraic expressions.

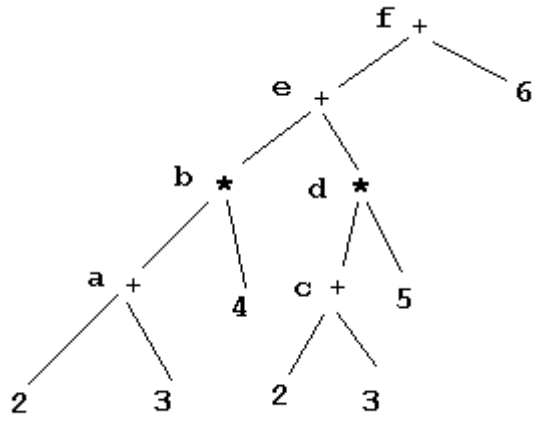
Complex algebraic expressions consisting of integers, addition and multiplication operators and parentheses can be represented by the following BNF notations

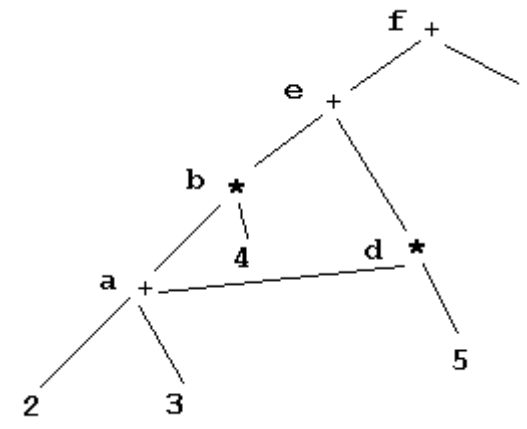
$< expr >$	$\rightarrow$	$< expr > + < term >$
$< expr >$	$\rightarrow$	$< term >$
$< term >$	$\rightarrow$	$< term > * < factor >$
$< term >$	$\rightarrow$	$< factor >$
$< factor >$	$\rightarrow$	$(< expr >)$
$< factor >$	$\rightarrow$	**num**

According to the above grammar, both operators are left associative and multiplication is given higher precedence. So, $2 + 3 * 4$ is same as $2 + (3 * 4)$. Similarly, $2 + 3 + 4$ is same as $(2+3) + 4$. On the other hand, $(2 + 3)$ * 4 is different to $2 + 3 * 4$. It may be surprising but $2 + 3 + 4$ is different to $2 + (3 + 4)$ as in this case, the order of evaluation is different.-

$2 + 3 + 4$	$2 + (3 + 4)$
$= 5 + 4$	$= 2 + 7$
$= 9$	$= 9$

Consider the expression $(2 + 3) * 4 + (2 + 3) * 5 + 6$





The above picture depicts two different ways of representing the expression in hand if we follow the given grammar. The left one is known as the AST (Abstract Syntax Tree) and the other one is DAG (common sub-expression is handled once). Such a representation clearly shows the way of evaluating an algebraic expression. Here, each non-leaf node is assigned a **unique** variable name. As a result, we can have a sequence of simple expressions to evaluate the expression

$a = 2 + 3$	$a = 2 + 3$
$b = a * 4$	$b = a * 4$
$c = 2 + 3$	$d = a * 5$
$d = c * 5$	$e = b + d$
$e = b + d$	$f = e + 6$
$f = e + 6$	

In both cases, f denotes the original expression $(2 + 3) * 4 + (2 + 3) * 5 + 6$. Here, you have to integrate a sequence of simple expressions into the original one. A simple expression will always be of the form

$$variable = (variable|integer)(+|*)(variable|integer)$$

For a given sequence of simple expressions, there can be several original expressions. Consider the following cases

$a = 2 + 3$	$b = (2 + 3) * 4$
$b = a * 4$	$b = ((2 + 3) * 4)$
Then b can be -	$b = ((2) + (3)) * 4$
	etc.

We need the first one, i.e. expression having minimum number of literals.

Input

The input file will start with an integer, T ($1 \leq T \leq 100$) denoting the number of tests. Each input will start with a positive integer, N ($1 \leq N \leq 50$) which is the number of simple expressions. Subsequent lines will contain the simple expressions defined above. The input will be valid, i.e. it will be always possible to construct a correct expression. Also, variable names will be referenced once they have been assigned. Either variable name or integers can have at most 10 characters. The integers will be always positive here (no unary minus) and variable names will contain letters only. Spaces will be used to separate numbers, variables and operators.

Output

For each input, print '`Expression #`D`:`' followed by the expression denoted by the last variable in the list of simple expressions. Here D is the test number, starting from 1. The length of any expression will not be greater than 5000.

Sample Input

```
2
2
A = 2 + 3
B = A + A
3
A = 2 + 3
B = A + 4
C = B + 5
```

Sample Output

```
Expression #1: 2+3+(2+3)
Expression #2: 2+3+4+5
```