Kick Start 2021 - Round C

Binary Operator

Problem

You are given a list of valid arithmetic expressions using non-negative integers, parentheses `()`, plus `+`, multiply `*`, and an extra operator `#`. The expressions are fully parenthesized and in infix notation.

A fully parenthesized expression is an expression where every operator and its operands are wrapped in a single parenthesis.
For example, the expression $x+y$ becomes $(x+y)$ when fully parenthesized, and $x+y+z$ becomes $((x+y)+z)$. However, 0 is still 0 when fully parenthesized, because it consists of a single number and no operators. $((x+y))$ is not considered fully parenthesized because it has redundant parentheses.

The operators $+$ and $*$ denote addition and multiplication, and $\#$ can be any total function.

You want to group the expressions into equivalence classes, where expressions are in the same equivalence class if and only if they are guaranteed to result in the same numeric value, regardless of which function $\#$ represents.

You can assume that $\#$ represents the same function across all expressions in a given test case. That might mean that $\#$ represents some known function like addition *or* subtraction, but not both in different parts of the same test case.

For example, consider the following expressions:

$$F_1\texttt{=((1\#(1+1))+((2\#3)*2))}$$
$$F_2\texttt{=(((2\#3)+(1\#2))+(2\#3))}$$
$$F_3\texttt{=((2*(2\#3))+(1\#2)).}$$

Let `A = 1#2`, and let `B = 2#3`. Then we can say F_1=F_2=F_3, regardless of the function $\#$ represents because the expressions can be rewritten as:

$$F_1\texttt{=((1\#2)+((2\#3)*2))=(A+(B*2))=(A+2B)}$$
$$F_2\texttt{=(((2\#3)+(2\#3))+(1\#2))=((B+B)+A)=(A+2B)}$$
$$F_3\texttt{=((2*(2\#3))+(1\#2))=((2*B)+A)=(A+2B).}$$

However, consider the expressions F_4`=((0#0)+(0#0))` and F_5`=(0#0)`. If $\#$ represents addition, then $F_4=F_5$. However, if $\#$ is $f(x,y)=C$, such that C is a non-zero integer, then $F_4 \neq F_5$ since $2C \neq C$. Therefore F_4 and F_5 are not in the same equivalence class.

Input

The first line of the input gives the number of test cases, $\mathbf{T}$. $\mathbf{T}$ test cases follow. Each test case begins with a line containing the integer $\mathbf{N}$. $\mathbf{N}$ lines follow. i-th line contains one expression, $\mathbf{E_i}$.

Output

For each test case, output one line containing `Case #x:` $Y_1, Y_2, \ldots, Y_\mathbf{N}$, where x is the test case number (starting from 1) and Y_i is the lexicographically smallest sequence satisfying the conditions below:

1. $1 \leq Y_i \leq Z$, where Z denotes the total number of equivalence classes in a given test case.
2. $Y_i = Y_j$ if and only if $\mathbf{E_i}$ and $\mathbf{E_j}$ are in the same equivalence class.

Limits

Time limit: 20 seconds.
Memory limit: 1 GB.
$1 \leq \mathbf{T} \leq 100$
$1 \leq \mathbf{N} \leq 100$
The length of $\mathbf{E_i}$ is at most 100, for all i.
$\mathbf{E_i}$ will be valid, for all i.

Test Set 1

No more than one `#` in each expression.

Test Set 2

No additional constraints.

Sample

Note: there are additional samples that are not run on submissions down below.

Sample Input

```
3
7
(1*(1#2))
(0*(1#2))
(1#2)
0
(3*0)
((1#2)*1)
(((1+(1#2))+3)*0)
5
(1*((1+(2#2))+3))
((0+(2#2))+4)
(100#2)
(((1+(2#2))+3)*1)
((50*2)#2)
2
(9999999999999999999999999999999
(100000000000000000000*100000000
```

Sample Output

```
Case #1: 1 2 1 2 2 1 2
Case #2: 1 1 2 1 2
Case #3: 1 1
```

This sample test set contains 3 test cases.

Test case 1 has 7 expressions and a total of 2 equivalence classes, denoted G_1 and G_2.

$\mathbf{E_1}$=`(1*(1#2))`, $\mathbf{E_2}$=`(0*(1#2))`, ..., $\mathbf{E_7}$=`(((1+(1#2))+3)*0)`.
$\mathbf{E_1}$, $\mathbf{E_3}$, and $\mathbf{E_6}$ belong to G_1, and $\mathbf{E_2}$, $\mathbf{E_4}$, $\mathbf{E_5}$, and $\mathbf{E_7}$ belong to G_2.
There are 2 sequences of Y_i that satisfy the requirement about equivalence classes in test case 1: `2 1 2 1 1 2 1` and `1 2 1 2 2 1 2`.

Since `1 2 1 2 2 1 2` is the lexicographically smaller one, the output for test case 1 is: `Case #1: 1 2 1 2 2 1 2`.

Test case 2 has 5 expressions and a total of 2 equivalence classes, denoted G_1 and G_2. $\mathbf{E_1}$, $\mathbf{E_2}$, and $\mathbf{E_4}$ belong to G_1, and $\mathbf{E_3}$ and $\mathbf{E_5}$ belong to G_2.
Therefore, the output for test case 2 is: `Case #2: 1 1 2 1 2`.

Test case 3 has 2 expressions that do not contain any `#`.
These two expressions evaluate to the same value, and therefore belong to the same equivalence class.

Additional Sample - Test Set 2

The following additional sample fits the limits of Test Set 2. It will not be run against your submitted solutions.

Sample Input

```
1
9
((2*(2#3))+(1#2))
(0*(1#2))
0
((1#(1+1))+((2#3)*2))
(3*0)
(1#(2#3))
(((2#3)+(1#2))+(2#3))
(4#7)
(7#4)
```

Sample Output

```
Case #1: 1 2 2 1 2 3 1 4 5
```

In the provided sample, there are a total of 5 equivalence classes. The first expression in the input is `((2*(2#3))+(1#2))`. All expressions from its equivalence class are denoted with `1` in the output. The equivalence class denoted with `2` consists of `(0*(1#2))`, `0`, and `(3*0)`. The equivalence class denoted with `3` consists of `(1#(2#3))`. Finally, the last two expressions, `(4#7)` and `(7#4)`, are not equivalent to any of the prior expressions or to one another. Note that `2 1 1 2 1 3 2 5 4` is one of many other sequences that satisfy the requirement about equivalence classes the given input, but it is not a correct answer because this sequence is not the lexicographically smallest one.