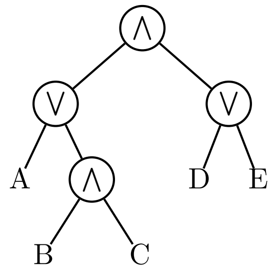
[C] Normalized Form

Program:	`tree.(c|cpp|java)`
Input:	`tree.in`
Balloon Color:	`Blue`

Description

As you most probably know, any boolean expression can be expressed in either a *disjunctive normal form* or a *conjunctive normal form.* In a disjunctive normal form, a boolean expression is written as a disjunct (logical or) of one-or more sub-expressions where each of these sub-expressions is written in a conjunctive normal form. Similarly, an expression written in a conjunctive normal form is a conjunct (logical and) of sub-expressions each written in a disjunctive normal form.

An AND/OR tree is a tree-like graphical-representation of boolean expressions written as either conjunctive- or disjunctive-normal form. Since the sub-expressions of a normalized form alternate in being either disjunctive or conjunctive forms, you'd expect the sub-trees on an AND/OR tree to alternate in being AND- or OR- trees depending on the sub-tree's depth-level. The example on the right illustrates this observation for the boolean expression $(A \bigvee (B \bigwedge C)) \bigwedge (D \bigvee E)$ where the trees in the 1st (top-most) and 3rd levels are AND-trees.



Write a program that evaluates a given and/or tree.

Input Format

Your program will be tested on one or more test cases. Each test case is specified on exactly one line (which is no longer than 32,000 characters) of the form:

$$(E_1 \; E_2 \; \ldots \; E_n)$$

where $n > 0$ and E_i is either `T` for true, `F` for false, or a sub-expression using the same format. The trees at the deepest level are AND-trees. The last test case is followed by a dummy line made of ().

Output Format

For each test case, print the following line:

```
k.␣E
```

Where `k` is the test case number (starting at one,) and `E` is either `true` or `false` depending on the value of the expression in that test case.

Sample Input/Output

tree.in
```
((F(TF))(TF))
(TFT)
((TFT)T)
()
```

OUTPUT
```
1. false
2. false
3. true
```