code jam
cout << "hello, world!" << endl;

Practice Mode

Round 1B 2009

A. Decision Tree

B. The Next Number

C. Square Math

Contest Analysis

Questions asked 2

- Submissions

Decision Tree

10pt	Not attempted **1512/1752 users** correct (86%)
11pt	Not attempted **1266/1544 users** correct (82%)

The Next Number

9pt	Not attempted **2559/3329 users** correct (77%)
26pt	Not attempted **1890/2557 users** correct (74%)

Square Math

12pt	Not attempted **157/422 users** correct (37%)
32pt	Not attempted **69/168 users** correct (41%)

- Top Scores

ACRush	100
ftc	100
bmerry	100
andrewzta	100
ipknHama	100
halyavin	100
mystic	100
Yarin	100
Khuc.Anh.Tuan	100
dgozman	100

Problem A. Decision Tree

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input
10 points

Download A-small-practice.in

your output file: Choose File No file chosen

source file(s): not needed for the practice contest

Submit file Hide

Large input
11 points

Solve A-large

Problem

Decision trees -- in particular, a type called classification trees -- are data structures that are used to classify *items* into *categories* using *features* of those items. For example, each animal is either "cute" or not. For any given animal, we can decide whether it is cute by looking at the animal's features and using the following decision tree.

```
(0.2 furry
  (0.81 fast
    (0.3)
    (0.2)
  )
  (0.1 fishy
    (0.3 freshwater
      (0.01)
      (0.01)
    )
    (0.1)
  )
)
```

A decision tree is defined recursively. It always has a root node and a weight. It also, optionally, has a feature name and two sub-trees, which are themselves decision trees.

More formally, a decision tree is defined using the following grammar.

```
tree ::= (weight [feature tree tree])
weight is a real number between 0 and 1, inclusive
feature is a string of 1 or more lower case English letters
```

The part inside the square brackets, **[]**, is optional. The parentheses, (), *weight* and *feature* are tokens. There will be at least one whitespace character between any two tokens, except (possibly) after an open-bracket '(' or before a close-bracket ')'. Whitespace characters are space (' ') and endline ('\n').

To figure out how likely the animal is to be cute, we start at the root of the tree with probability *p* set to 1. At each node, we multiply *p* by the weight of the node. If the node is a leaf (has no sub-trees), then we stop, and the value of *p* is the probability that our animal is cute. Otherwise, we look at the feature associated with the node. If our animal has this feature, we move down into the first sub-tree and continue recursively. If it does not have this feature, then we move down into the second sub-tree and continue in the same way.

For example, a beaver is an animal that has two features: *furry* and *freshwater*. We start at the root with *p* equal to 1. We multiply *p* by 0.2, the weight of the root and move into the first sub-tree because the beaver has the *furry* feature. There, we multiply *p* by 0.81, which makes *p* equal to 0.162. From there we move further down into the second sub-tree because the beaver does not have the *fast* feature. Finally, we multiply *p* by 0.2 and end up with 0.0324 -- the probability that the beaver is cute.

You will be given a decision tree and a list of animals with their features. For each item, you need to return the probability that the animal is cute.

Input

The first line of input contains a single integer, **N**, the number of test cases. **N** test cases follow.

Each test case description will start with a line that contains an integer **L** -- the number of lines that describe a decision tree. The next **L** lines will contain a decision tree in the format described above. The line after that will contain **A** -- the number of animals. The next **A** lines will each contain the description of one animal in the following format.

$$animal\ \mathbf{n}\ feature_1\ feature_2\ \ldots\ feature_{\mathbf{n}}$$

Output

For each test case, output one line containing "Case #**x**:" followed by exactly **A** lines, one per animal, in the same order as they appear in the input. Each line should contain the probability that the animal is cute. Answers that are precise to within an absolute or relative error of 10^{-6} will be considered correct.

Limits

$1 \le \mathbf{N} \le 100$
All weights will be between 0 and 1, inclusive.
All weights will consist of only digits with at most one decimal point.
The weights will not start or end with a decimal point.
The weights will not have more than one 0 before a decimal point.
All animals and features will consist of between 1 and 10 lower case English letters.
All animal names within a test case will be distinct.
All feature names for a single animal will be distinct.
Each of the **L** lines in a decision tree definition will have at most 80 characters, not including the endlines.

Small dataset

$1 \le \mathbf{L} \le 10$
$1 \le \mathbf{A} \le 10$
$0 \le \mathbf{n} \le 5$

Large dataset

$1 \le \mathbf{L} \le 100$
$1 \le \mathbf{A} \le 100$
$0 \le \mathbf{n} \le 100$

Sample

Input

```
2
3
(0.5 cool
  ( 1.000)
  (0.5 ))
2
anteater 1 cool
cockroach 0
13
(0.2 furry
  (0.81 fast
    (0.3)
    (0.2)
  )
  (0.1 fishy
    (0.3 freshwater
      (0.01)
      (0.01)
    )
    (0.1)
  )
)
3
beaver 2 furry freshwater
trout 4 fast freshwater fishy rainbowy
dodo 1 extinct
```

Output

```
Case #1:
0.5000000
0.2500000
Case #2:
0.0324000
0.0000600
0.0020000
```