2518 Random Walk

Random Algorithms are very popular now. For example, it's widely used in cryptography (primality test and etc.).

Different from the normal definite algorithms, random algorithms may not have a definite running trace. Sometimes, it comes to some point, flips a coin, and decides where to go according to the result of the coin flipping.

The following command may be such a point:

```
x=random(1); // random(1) returns a random number from [0..1) uniformly

IF x > 0.3 GOTO . . .
```

When the random program comes here, it generates a random number and assigns it to x, if x is greater than 0.3 it goes to somewhere; otherwise, it goes straight ahead. So, the program goes somewhat like a drunk man walking a long the street. You can't predict where the program should go beforehand.

Your task in this problem is to create a simple Random Program Evaluator, which can calculate the expected running time of some random algorithms.

A simple random program is described as follows:

1. There are several reserved words: "`NOP`", "`IF`", "`GOTO`", "`END`", "`PROC`", "`PROG_START`", and "`PROG_END`", case is sensitive.
2. A program starts with "`PROG_START`" and ends with "`PROG_END`".
3. A program may have one or more procedures.
4. Each procedure starts with "`PROC` [*name*]". [*name*] is the name of this procedure. It can be any string that only contains characters or numbers, except reserved words.
5. A procedure ends with "`END;`"
6. A procedure may have one or more commands ("`END;`" is not a command). It takes one time unit to execute one command.
7. A command is of one of the following formats:
   - "`NOP;`" The program will go to execute the next line in the next time unit.
   - "`IF x>`[*threshold*] `GOTO` [*line number*]`;`". The program takes a value x from [0..1) randomly. If x is larger than the [*threshold*], the program will execute the [*line number*] in the next time unit, otherwise, the program execute the next line in the next time unit.
   - "`IF x<`[*threshold*] `GOTO` [*line number*]`;`". The program takes a value x from [0..1) randomly. If x is smaller than the [*threshold*], the program will execute the [*line number*] in the next time unit, otherwise, the program execute the next line in the next time unit.
   - "`IF x>`[*threshold*] `PROC` [*procedure name*]`;`". The program takes a value x from [0..1) randomly. If x is larger than the [*threshold*], the program will execute the procedure with [*procedure name*], then come back to the next line after the current command in this procedure.

- "`IF x<`[*threshold*] `PROC` [*procedure name*]`;`". The program takes a value x from [0..1) randomly. If x is smaller than the [*threshold*], the program will execute the procedure with [*procedure name*], then come back to the next line after the current command in this procedure.

8. The program finishes one procedure whenever it sees the "`END;`".
9. In each procedure, the line number starts with 1. That is, the first command in a procedure is "line 1", the second is "line 2". "`END;`" is the last line in a procedure.

The Evaluator takes one random program as input; return the expected runtime of some procedures upon requests, with an accuracy of 3 digits after the decimal point.

To make your lives easier, we have the following simplification:

1. The condition of "`IF`" is always a comparison between a random variable and a constant threshold.
2. Random Variables in different "`IF`" clause are independent. Even there may be more than one "`IF x>0.5`" in one procedure, the 'x's in different commands are independent.
3. There is no loop-reference (including indirect loop reference) between different procedures.
4. For each command, the probability that the program starts from this command and ends at the end of the procedure is positive. That means, you can finally reach the end of a procedure from any command.
5. There are more than 1 and less than 100 procedures in a program
6. All the strings in this problem are within 100 characters.

Input

The input file contains one random program described as above with one or more random procedures, and one or more requests. The requests are following the program. Each request is a line with one string, which is the name of the requested procedure. The request list ends with a line containing "`REQUEST_END`" (there is no procedure with a name as "`REQUEST_END`")

Output

For each request, output a line with a number presents the expected running time of the requested procedure. The accuracy is up to 3 digits after decimal point.

Sample Input

```
PROG_START
PROC A
IF x>0.5 GOTO 3;
NOP;
END;
PROC B
IF x<0.5 PROC A;
NOP;
END;
PROG_END
B
A
REQUEST_END
```

Sample Output

```
2.750
1.500
```