UVa Online Judge

11619 SPAM! (or not)

Since Paul Graham wrote his famous article “A Plan for Spam,” people started to use Bayesian filters to classify incoming mail as spam or ham. This form of classification relies on previous learning from known spam or ham (non-spam) messages.

The learning algorithm assigns probabilities to words appearing in messages that are already classified by the user.

Thus, using the Bayes theorem, we can say that the probability of a message being spam when we know that this message contains the word w is:

$$P(S|w) = \frac{P(w|S) \times P(S)}{P(w|S) \times P(S) + P(w|H) \times P(H)}$$

where the probabilities $P(w|S)$ and $P(w|H)$ are calculated by the previous learning phase of the algorithm. If we suppose that there is equal possibility of being spam and ham (50%), then the formula gets simplified as:

$$P(S|w) = \frac{P(w|S)}{P(w|S) + P(w|H)}$$

However, as we have a small learning set, we are going to limit a little bit the range of probabilities (also to make the algorithm a little bit more fuzzy), so instead of $P(S|w)$ we will use $P'(S|w)$, defined as:

$$p(x) = \begin{cases} 0.01 & \text{if} \quad P(S|w) < 0.01 \\ 0.99 & \text{if} \quad P(S|w) > 0.99 \\ P(S|w) & \text{otherwise} \end{cases}$$

A message can be seen as composed as a series of words. If we consider each word independent, we can obtain a function to calculate the probability (called p) of a message to be spam:

$$p = \frac{\prod\limits_{w \in message} P'(S|w)}{\prod\limits_{w \in message} P'(S|w) + \prod\limits_{w \in message} (1 - P'(S|w))}$$

where w represents each word in the message.

Finally, there is only one special case when calculating $P(S|w)$ when w has not been seen either in spam or ham. Then, as Graham estimates, a probability of 40% (0.4) is given, that is:

$$P(S|w) = 0.4 \quad \forall w | w \notin spamwords \land w \notin hamwords$$

where *spamwords* and *hamwords* are the set of spam and ham words found in spam and ham messages respectively.

As an assignment, you have to write a program that is able to learn from known messages being either spam or ham, and also to classify other messages given as input based on their probability to be spam.

Input

The input will be composed of a series of messages. Each message has a first line (a header), then several lines with a series of words each (separated by one or more spaces), and ends with a line containing ‘==’ by itself. The words will be composed of letters, that may be in mixed case. You have to convert them all to lower case.

As for the header, it can have three possible values:

1. `MESSAGE SPAM` (a message known to be spam.)
2. `MESSAGE HAM` (a message known to be non-spam or ham.)
3. `MESSAGE CLASSIFY` (the message has to be classified as spam or ham.)

Types 1 and 2 represent the learning process for the algorithm, while type 3 requires the program to produce an output classifying the message as either spam or ham. The types of messages can be intermixed in the input.

Output

For each message with the header '`MESSAGE CLASSIFY`', the program will produce a line with three possible values depending on the probability (p) of the message being spam:

- `Spam` if $p\ ge 0.6$.
- `Ham` if $p \leq 0.4$.
- `Unsure` if $p \in (0.4, 0.6)$.

Sample Input

```
MESSAGE SPAM
replica rolex
==
MESSAGE SPAM
replica rolex
==
MESSAGE HAM
Hello all
==
MESSAGE CLASSIFY
Hello replica
==
MESSAGE CLASSIFY
replica rolex and some other things
==
MESSAGE CLASSIFY
abc def ghi jkl all
==
```

Sample Output

```
Unsure
Spam
Ham
```