

Unsolicited email (spam) is annoying and clutters your mailbox. You are to write a *spam filter* - a program that reads email messages of regular ASCII characters (32-127) and tries to determine whether or not each message is spam.

How can we determine whether or not a message is spam? Spam contains words and phrases that are not common in genuine email messages. For example, the phrase

MAKE MONEY FAST, HONEY!!

is in all-uppercase, contains the word "money" and ends with a double exclamation mark.

One way to create a spam filter is to read through many spam and non-spam messages and to come up with a set of rules that will classify any particular message as spam or not. This process can be tedious and error prone to do manually. Instead you will write a program to automate the process.

A useful step in automatic classification is to split the text up into set of *trigrams*. A trigram is a sequence of three adjacent characters that appear in the message. A trigram is case sensitive. The example above is composed of the trigrams:

MAK
AKE
KE
E M
MO
MON
ONE
NEY
EY
Y F
FA
FAS
AST
ST,
T,
, H
HO
HON
ONE
NEY
EY!
Y!!

If we examine a sample of spam and non-spam messages we find that some trigrams are more common in spam; whereas others are more common in non-spam. This observation leads to a classification method:

- Examine a sample consisting of a large number of spam messages. Count the number of times that each trigram occurs. In the example above, there are 20 distinct trigrams; the trigrams ONE and NEY occur twice each and the remaining 18 trigrams occur once each. (Trigrams that do not occur are considered to occur 0 times.) More formally, for each trigram t we compute the frequency $f_{spam}(t)$ with which it occurs in the sample of spam.
- Examine a sample consisting of a large number of non-spam messages. Compute $f_{non-spam}(t)$, the frequency with which each trigram t appears in the sample of non-spam.
- For each message to be filtered, compute $f_{message}(t)$ for each trigram t .
- If $f_{message}$ resembles f_{spam} more closely than it resembles $f_{non-spam}$ it is determined to be spam; otherwise it is determined to be non-spam.
- A similarity measure determines how closely f_1 and f_2 resemble one another. One of the simplest measures is the cosine measure:

$$similarity(f_1, f_2) = \frac{\sum_t f_1(t) \cdot f_2(t)}{\sqrt{\sum_t [f_1(t)]^2} \cdot \sqrt{\sum_t [f_2(t)]^2}}$$

Then we say that a message is spam if

$$similarity(f_{message}, f_{spam}) > similarity(f_{message}, f_{non-spam})$$

Input

The input file contains several sets (less than 10) of input. The description of each set is given below.

The first line of each set contains three integers: $s(0 < s < 5)$ the number of sample spam messages to follow; $n(0 < n < 5)$ the number of sample non-spam messages to follow; $c(0 < c < 10)$ the number of messages to be classified as spam or non-spam, based on trigram the trigram frequencies of the sample messages. Each message consists of several lines of text and is terminated by a line containing 'ENDMESSAGE'. This line will not appear elsewhere in the input, and is not considered part of the message. No line has more than 1000 characters.

Input is terminated by a set where $s = 0$, $n = 0$ and $c = 0$. This set should not be processed.

Output

For each set of input your program should output a single line to identify the serial of the input set. The output specification for each set is given below:

For each of the c messages, your program will output two lines. On the first line, output $similarity(f_{message}, f_{spam})$ and $similarity(f_{message}, f_{non-spam})$. On the second line print the classification of the message ('spam' or 'non-spam'). Round the numbers to five decimal digits.

For detailed description look at the output for sample input.

When forming trigrams, we never include a new line character. We don't include trigrams that span multiple lines, either.

Sample Input

```
2 1 1
AAAA
BBBB CCCC
ENDMESSAGE
BBBB
ENDMESSAGE
AAAABBBB
ENDMESSAGE
AAABB
ENDMESSAGE
2 1 2
AAAA
BBBB CCCC
ENDMESSAGE
BBBB
ENDMESSAGE
AAAABBBB
ENDMESSAGE
AAABB
ENDMESSAGE
AAABB
ENDMESSAGE
0 0 0
```

Sample Output

```
Set 1:
0.21822 0.73030
non-spam
Set 2:
0.21822 0.73030
non-spam
0.21822 0.73030
non-spam
```