A. Expecting Trouble

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

It is Friday the 13th today, and even though you're a modern well-educated person, you can't help feeling a bit nervous about it. You decide to look for evidence against this superstition (or for it). As a first step, you recall all Fridays the 13th in your life and calculate how many of them were unusually bad — like that time when you decided to play a game on ZX Spectrum and ended up breaking your TV set. The problem is, you can't remember some Fridays, and you're not sure why — were they really that bad?

You have assembled a sequence of your recollections. Character "0" stands for a normal day, "1" — for a nasty one, and "?" means you have no idea what kind of day that was. Being a programmer, you decide to approximate these unknown days with independent random variables, which take value 1 with probability p, and 0 with probability $(1 - p)$.

Given a string of your memories and the value of p, calculate out the expected value of average badness of your Fridays the 13th.

Input

The first line of the input contains a string s which represents your Fridays; s will contain between 1 and 50 characters, inclusive. Each character of s will be "0", "1" or "?".

The second line of the input contains a double p $(0 \le p \le 1)$. Double p is given with at most 2 digits after the decimal point.

Output

Output the expected value of average badness of your Fridays with exactly 5 decimal places. Please, use standard mathematical rules when you are rounding an answer.

Examples

input

```
?111?1??1
1.0
```

output

```
1.00000
```

input

```
01?10??10000
0.5
```

output

```
0.37500
```

Note

In the first case, you're doomed. DOOMED! Sorry, just had to say that.

B. Triskaidekaphobia

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Triskaidekaphobia is a fear of number 13. Ordinary people who suffer from this phobia feel uncomfortable around numbers 13, 130, 513 etc, but you, being a programmer, take this fear one step further. For example, consider number 7. It's ok when written in decimal, but written in base 4 it becomes 13, the dreadful number!

The more you think about it, the worse it looks. Number 100 has as many as 13 notations which contain 13! And there are numbers which have 13 in infinite number of their notations! Luckily, you can do any math in binary, which is completely safe from the nasty number. But still, you want to be able to estimate the level of nastiness of any number.

Your task is: given an integer n, find the number of different integer bases b $(b \geq 2)$ for which n, written in base b, contains at least one 13. Assume that "digits" of the number in bases larger than 10 are written not as letters but as decimal numbers; thus, 30 in base 16 is not 1E but (1)(14) or simply 114. Please note, that 13 must be present as a substring of notation, not a subsequence (123 doesn't contain 13).

Input

The only line of the input contains a single integer n $(1 \leq n \leq 10^5)$.

Output

Output a single integer — the number of different integer bases b $(b \geq 2)$ for which n, written in base b, contains at least one 13. If there are infinitely many such bases, output -1.

Examples

input
```
7
```
output
```
1
```

input
```
100
```
output
```
13
```

input
```
13
```
output
```
-1
```

C. Counting Fridays

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Judging from the previous problem, Friday the 13th really doesn't treat you well, so you start thinking about how to minimize its impact on your life. You are a passionate participant of programming contests, so various competitions are an important part of your schedule. Naturally, you'd like as few of them to be spoiled as possible.

A friendly admin leaked to you the list of dates on which the contests will be held for several years ahead. Check how many of them happen to take place on Friday the 13th of any month.

Input

The first line of the input contains an integer n $(1 \le n \le 10)$ — the number of the contests you have been told about. The following n lines contain dates of these contests, one per line, formatted as "`YYYY-MM-DD`" (1974 $\le$ `YYYY` $\le$ 2030; 01 $\le$ `MM` $\le$ 12; 01 $\le$ `DD` $\le$ 31). It's guaranteed that all the given dates are correct. Two distinct contests may be at the same day.

Output

Output a single integer — the number of dates which happen to be Friday the 13th.

Examples

input

```
5
2012-01-13
2012-09-13
2012-11-20
2013-09-13
2013-09-20
```

output

```
2
```

D. Chain Letter

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

A chain letter is a kind of a message which urges the recipient to forward it to as many contacts as possible, usually with some kind of mystic explanation. Of course, this is only a superstition, and you don't believe in it, but all your friends do. You know that today there will be one of these letters going around, and you want to know how many times you'll receive it — of course, not that you'll be sending it yourself!

You are given an array of strings f with n elements which describes the contacts between you and $n - 1$ of your friends: j-th character of i-th string ($f[i][j]$) is "1" if people i and j will send messages to each other, and "0" otherwise. Person 1 starts sending the letter to all his contacts; every person who receives the letter for the first time sends it to all his contacts. You are person n, and you don't forward the letter when you receive it.

Calculate the number of copies of this letter you'll receive.

Input

The first line of the input contains an integer n $(2 \leq n \leq 50)$ — the number of people involved. Next n following lines contain elements of f, strings of length n. Each character in f is either "0" or "1". It's guaranteed that two following equations hold: $f[i][j] = f[j][i]$, $f[i][i] = 0$, for all i, j $(1 \leq i, j \leq n)$.

Output

Output a single integer — the number of copies of the letter you will receive eventually.

Examples

input

```
4
0111
1011
1101
1110
```

output

```
3
```

input

```
4
0110
1010
1100
0000
```

output

```
0
```

input

```
4
0101
1001
0001
1110
```

output

```
2
```

Note

In the first case, everybody sends letters to everyone, so you get copies from all three of your friends.

In the second case, you don't know any of these people, so they don't bother you with their superstitious stuff.

In the third case, two of your friends send you copies of the letter but the third friend doesn't know them so he is unaffected.

E. Black Cat Rush

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Today you go out of your house and immediately notice that something is weird. Around your door there is a swarm of black cats — all tense paws and twitching tails. As you do your first step, they all dart off and start running towards you. It looks like they want to thwart you!

You are moving in a straight line from point $(0, 0)$ to point $(a, 0)$ with velocity v. There are n black cats around you who want to cross your paths. A cat can move in any direction with velocity at most u. A cat assumes it has crossed your path if it managed to get to at least one point of your path earlier or at the same time as you got there.

You are given four integers: a, v, u, n, as well as cats' coordinates (x_i, y_i). What is the greatest number of cats who manage to cross your path?

Input

The first line contains four integers a, v, u, n $(1 \le a \le 10000;\ 1 \le v, u \le 100;\ 1 \le n \le 1000)$. Each of the next n lines contains two integers x_i, y_i $(-100 \le x_i, y_i \le 100)$ — location of the i-th cat.

It's guaranteed that all cats are located in distinct points.

Output

Output a single integer — what is the greatest number of cats who manage to cross your path?

Examples

input

```
1 1 5 4
0 3
4 -4
7 0
-2 -2
```

output

```
3
```

input

```
10 5 3 4
7 5
5 2
10 -7
15 0
```

output

```
3
```

F. Superstitions Inspection

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You read scientific research regarding popularity of most famous superstitions across various countries, and you want to analyze their data. More specifically, you want to know which superstitions are popular in most countries.

The data is given as a single file in the following format: country name on a separate line, followed by a list of superstitions popular in it, one entry per line. Each entry of the list is formatted as an asterisk "*" followed by a single space and then by the superstition name. Entries are unique within each country. The first line of input will represent a country. The input will contain at least one superstition.

Country name is a non-empty sequence of words, where each word consists of only English letters. The words of the name of some country will be separated in input with one or more spaces.

Superstition name is a non-empty sequence of words, where each word consists of only English letters and digits. The words of the name of some superstition will be separated in input with one or more spaces.

You can consider two names equal, if corresponding sequences of words are equal. You shouldn't consider the case of the letters when you compare the words.

Output the list of superstitions which are observed in the greatest number of countries. It's guaranteed that all countries have distinct names.

Input

The input contains between 2 and 50 lines. Every line of input will contain between 1 and 50 characters, inclusive.

No line has leading or trailing spaces.

Output

Output the list of superstitions which are observed in the greatest number of countries in alphabetical order. Each superstition must be converted to lowercase (one superstition can be written with varying capitalization in different countries).

The alphabetical order of superstitions means the lexicographical order of sequences of words (their names).

Examples

input

```
Ukraine
* Friday the   13th
* black   cat
* knock the   wood
USA
* wishing well
* friday   the   13th
Holland
France
* Wishing Well
```

output

```
friday the 13th
wishing well
```

input

```
Spain
* Tuesday the 13th
Italy
* Friday the 17th
Russia
* Friday the 13th
England
* rabbit foot
```

output

```
friday the 13th
friday the 17th
rabbit foot
tuesday the 13th
```

G. Suffix Subgroup

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a group of n strings: $s_1, s_2, \ldots, s_n$.

You should find a subgroup $s_{i_1}, s_{i_2}, \ldots, s_{i_k}$ $(1 \le i_1 < i_2 < \ldots < i_k \le n)$ of the group. The following two conditions must hold:

- there exists a string t such, that each string from found subgroup is its suffix;
- the number of strings in the found subgroup is as large as possible.

Your task is to print the number of strings in the found subgroup.

Input

The first line contains an integer n $(1 \le n \le 10^5)$ — the number of strings in the group. Each of the next n lines contains a string. The i-th line contains non-empty string s_i.

Each string consists only from lowercase Latin letters. The sum of all strings s_i doesn't exceed 10^5.

Output

Output a single integer — the number of strings in the found subgroup.

Examples

input

```
6
bb
bb
b
aaa
aa
z
```

output

```
3
```

Note

Look at the test sample. The required subgroup is s_1, s_2, s_3.