A. Message

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

Dr. Moriarty is about to send a message to Sherlock Holmes. He has a string s.

String p is called a *substring* of string s if you can read it starting from some position in the string s. For example, string "aba" has six substrings: "a", "b", "a", "ab", "ba", "aba".

Dr. Moriarty plans to take string s and cut out some substring from it, let's call it t. Then he needs to *change* the substring t zero or more times. As a result, he should obtain a fixed string u (which is the string that should be sent to Sherlock Holmes). One change is defined as making one of the following actions:

- Insert one letter to any end of the string.
- Delete one letter from any end of the string.
- Change one letter into any other one.

Moriarty is very smart and after he chooses some substring t, he always makes the minimal number of changes to obtain u.

Help Moriarty choose the best substring t from all substrings of the string s. The substring t should minimize the number of changes Moriarty should make to obtain the string u from it.

Input

The first line contains a non-empty string s, consisting of lowercase Latin letters. The second line contains a non-empty string u, consisting of lowercase Latin letters. The lengths of both strings are in the range from 1 to 2000, inclusive.

Output

Print the only integer — the minimum number of changes that Dr. Moriarty has to make with the string that you choose.

Sample test(s)

input
```
aaaaa
aaa
```
output
```
0
```

input
```
abcabc
bcd
```
output
```
1
```

input
```
abcdef
klmnopq
```
output
```
7
```

Note

In the first sample Moriarty can take any substring of length 3, and it will be equal to the required message u, so Moriarty won't have to make any changes.

In the second sample you should take a substring consisting of characters from second to fourth ("bca") or from fifth to sixth ("bc"). Then you will only have to make one change: to change or to add the last character.

In the third sample the initial string s doesn't contain any character that the message should contain, so, whatever string you choose, you will have to make at least 7 changes to obtain the required message.

B. Suspects

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

As Sherlock Holmes was investigating a crime, he identified n suspects. He knows for sure that exactly one of them committed the crime. To find out which one did it, the detective lines up the suspects and numbered them from 1 to n. After that, he asked each one: "Which one committed the crime?". Suspect number i answered either "The crime was committed by suspect number a_i", or "Suspect number a_i didn't commit the crime". Also, the suspect could say so about himself ($a_i = i$).

Sherlock Holmes understood for sure that exactly m answers were the truth and all other answers were a lie. Now help him understand this: which suspect lied and which one told the truth?

Input

The first line contains two integers n and m ($1 \le n \le 10^5, 0 \le m \le n$) — the total number of suspects and the number of suspects who told the truth. Next n lines contain the suspects' answers. The i-th line contains either "+a_i" (without the quotes), if the suspect number i says that the crime was committed by suspect number a_i, or "-a_i" (without the quotes), if the suspect number i says that the suspect number a_i didn't commit the crime (a_i is an integer, $1 \le a_i \le n$).

It is guaranteed that at least one suspect exists, such that if he committed the crime, then exactly m people told the truth.

Output

Print n lines. Line number i should contain "`Truth`" if suspect number i has told the truth for sure. Print "`Lie`" if the suspect number i lied for sure and print "`Not defined`" if he could lie and could tell the truth, too, depending on who committed the crime.

Sample test(s)

input

```
1 1
+1
```

output

```
Truth
```

input

```
3 2
-1
-2
-3
```

output

```
Not defined
Not defined
Not defined
```

input

```
4 1
+2
-3
+4
-1
```

output

```
Lie
Not defined
Lie
Not defined
```

Note

The first sample has the single person and he confesses to the crime, and Sherlock Holmes knows that one person is telling the truth. That means that this person is telling the truth.

In the second sample there are three suspects and each one denies his guilt. Sherlock Holmes knows that only two of them are telling the truth. Any one of them can be the criminal, so we don't know for any of them, whether this person is telling the truth or not.

In the third sample the second and the fourth suspect defend the first and the third one. But only one is telling the truth, thus, the first or the third one is the criminal. Both of them can be criminals, so the second and the fourth one can either be lying or telling the truth. The first and the third one are lying for sure as they are blaming the second and the fourth one.

C. Cipher

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

Sherlock Holmes found a mysterious correspondence of two VIPs and made up his mind to read it. But there is a problem! The correspondence turned out to be encrypted. The detective tried really hard to decipher the correspondence, but he couldn't understand anything.

At last, after some thought, he thought of something. Let's say there is a word s, consisting of $|s|$ lowercase Latin letters. Then for one *operation* you can choose a certain position p $(1 \le p < |s|)$ and perform one of the following actions:

- either replace letter s_p with the one that alphabetically **follows** it and replace letter s_{p+1} with the one that alphabetically **precedes** it;
- or replace letter s_p with the one that alphabetically **precedes** it and replace letter s_{p+1} with the one that alphabetically **follows** it.

Let us note that letter "z" doesn't have a defined following letter and letter "a" doesn't have a defined preceding letter. That's why the corresponding changes are not acceptable. If the operation requires performing at least one unacceptable change, then such operation cannot be performed.

Two words *coincide in their meaning* iff one of them can be transformed into the other one as a result of zero or more operations.

Sherlock Holmes needs to learn to quickly determine the following for each word: how many words can exist that coincide in their meaning with the given word, but differs from the given word in at least one character? Count this number for him modulo 1000000007 $(10^9 + 7)$.

Input

The input data contains several tests. The first line contains the only integer t $(1 \le t \le 10^4)$ — the number of tests.

Next t lines contain the words, one per line. Each word consists of lowercase Latin letters and has length from 1 to 100, inclusive. Lengths of words can differ.

Output

For each word you should print the number of different **other** words that coincide with it in their meaning — not from the words listed in the input data, but from all possible words. As the sought number can be very large, print its value modulo 1000000007 $(10^9 + 7)$.

Sample test(s)

input

```
1
ab
```

output

```
1
```

input

```
1
aaaaaaaaaaa
```

output

```
0
```

input

```
2
ya
klmbfxzb
```

output

```
24
320092793
```

Note

Some explanations about the *operation*:

- Note that for each letter, we can clearly define the letter that follows it. Letter "b" alphabetically follows letter "a", letter "c" follows letter "b", ..., "z" follows letter "y".
- Preceding letters are defined in the similar manner: letter "y" precedes letter "z", ..., "a" precedes letter "b".
- Note that the operation never changes a word's length.

In the first sample you can obtain the only other word "ba". In the second sample you cannot obtain any other word, so the correct answer is 0.

Consider the third sample. One operation can transform word "klmbfxzb" into word "klmcexzb": we should choose $p = 4$, and replace the fourth letter with the following one ("b" $\rightarrow$ "c"), and the fifth one — with the preceding one ("f" $\rightarrow$ "e"). Also, we can obtain many other words from this one. An operation can transform word "ya" only into one other word "xb".

Word "ya" coincides in its meaning with words "xb", "wc", "vd", ..., "ay" (overall there are 24 other words). The word "klmbfxzb has many more variants — there are 3320092814 other words that coincide with in the meaning. So the answer for the first word equals 24 and for the second one equals 320092793 — the number 3320092814 modulo $10^9 + 7$

D. Clues

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

As Sherlock Holmes was investigating another crime, he found a certain number of clues. Also, he has already found *direct links* between some of those clues. The direct links between the clues are mutual. That is, the direct link between clues A and B and the direct link between clues B and A is the same thing. No more than one direct link can exist between two clues.

Of course Sherlock is able to find direct links between all clues. But it will take too much time and the criminals can use this extra time to hide. To solve the crime, Sherlock needs each clue to be linked to all other clues (maybe not directly, via some other clues). Clues A and B are considered linked either if there is a direct link between them or if there is a direct link between A and some other clue C which is linked to B.

Sherlock Holmes counted the minimum number of additional direct links that he needs to find to solve the crime. As it turns out, it equals T.

Please count the number of different ways to find exactly T direct links between the clues so that the crime is solved in the end. Two ways to find direct links are considered different if there exist two clues which have a direct link in one way and do not have a direct link in the other way.

As the number of different ways can turn out rather big, print it modulo k.

Input

The first line contains three space-separated integers n, m, k ($1 \le n \le 10^5$, $0 \le m \le 10^5$, $1 \le k \le 10^9$) — the number of clues, the number of direct clue links that Holmes has already found and the divisor for the modulo operation.

Each of next m lines contains two integers a and b ($1 \le a, b \le n$, $a \ne b$), that represent a direct link between clues. It is guaranteed that any two clues are linked by no more than one direct link. Note that the direct links between the clues are mutual.

Output

Print the single number — the answer to the problem modulo k.

Sample test(s)

input

```
2 0 1000000000
```

output

```
1
```

input

```
3 0 100
```

output

```
3
```

input

```
4 1 1000000000
1 4
```

output

```
8
```

Note

The first sample only has two clues and Sherlock hasn't found any direct link between them yet. The only way to solve the crime is to find the link.

The second sample has three clues and Sherlock hasn't found any direct links between them. He has to find two of three possible direct links between clues to solve the crime — there are 3 ways to do it.

The third sample has four clues and the detective has already found one direct link between the first and the fourth clue. There are 8 ways to find two remaining clues to solve the crime.

E. Mrs. Hudson's Pancakes

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Mrs. Hudson hasn't made her famous pancakes for quite a while and finally she decided to make them again. She has learned m new recipes recently and she can't wait to try them. Those recipes are based on n special spices. Mrs. Hudson has these spices in the kitchen lying in jars **numbered** with integers from 0 to $n - 1$ (each spice lies in an individual jar). Each jar also has the **price** of the corresponding spice inscribed — some integer a_i.

We know three values for the i-th pancake recipe: d_i, s_i, c_i. Here d_i and c_i are integers, and s_i is the *pattern* of some integer written in the numeral system with radix d_i. The pattern contains digits, Latin letters (to denote digits larger than nine) and question marks. Number x in the d_i-base numeral system *matches the pattern* s_i, if we can replace question marks in the pattern with digits and letters so that we obtain number x (leading zeroes aren't taken into consideration when performing the comparison). More formally: each question mark should be replaced by exactly one digit or exactly one letter. If after we replace all question marks we get a number with leading zeroes, we can delete these zeroes. For example, number `40A9875` in the 11-base numeral system matches the pattern "`??4??987?`", and number `4A9875` does not.

To make the pancakes by the i-th recipe, Mrs. Hudson should take all jars with **numbers** whose representation in the d_i-base numeral system matches the pattern s_i. The *control number* of the recipe (z_i) is defined as the sum of number c_i and the product of **prices** of all taken jars. More formally: $z_i = c_i + \prod_j a_j$ (where j is all such numbers whose representation in the d_i-base numeral system matches the pattern s_i).

Mrs. Hudson isn't as interested in the control numbers as she is in their minimum prime divisors. Your task is: for each recipe i find the minimum prime divisor of number z_i. If this divisor exceeds 100, then you do not have to find it, print -1.

Input

The first line contains the single integer n ($1 \le n \le 10^4$). The second line contains space-separated prices of the spices $a_0, a_1, ..., a_{n-1}$, where a_i is an integer ($1 \le a_i \le 10^{18}$).

The third line contains the single integer m ($1 \le m \le 3 \cdot 10^4$) — the number of recipes Mrs. Hudson has learned.

Next m lines describe the recipes, one per line. First you are given an integer d_i, written in the decimal numeral system ($2 \le d_i \le 16$). Then after a space follows the s_i pattern — a string from 1 to 30 in length, inclusive, consisting of digits from "`0`" to "`9`", letters from "`A`" to "`F`" and signs "`?`". Letters from "`A`" to "`F`" should be considered as digits from 10 to 15 correspondingly. It is guaranteed that all digits of the pattern (including the digits that are represented by letters) are strictly less than d_i. Then after a space follows an integer c_i, written in the decimal numeral system ($1 \le c_i \le 10^{18}$).

Please do not use the `%lld` specificator to read or write 64-bit integers in C++, in is preferred to use `cin`, `cout`, strings or the `%I64d` specificator instead.

Output

For each recipe count by what minimum prime number the control number is divided and print this prime number on the single line. If this number turns out larger than 100, print -1.

Sample test(s)

input
```
1
1
1
2 ? 1
```
output
```
2
```

input
```
4
2 3 5 7
4
2 ?0 11
2 ?1 13
2 0? 17
2 1? 19
```
output
```
3
2
23
2
```

input
```
1
1000000000000000000
1
16 ?????????????? 1
```
output

```
-1
```

Note

In the first test any one-digit number in the binary system matches. The jar is only one and its price is equal to 1, the number c is also equal to 1, the control number equals 2. The minimal prime divisor of 2 is 2.

In the second test there are 4 jars with numbers from 0 to 3, and the prices are equal 2, 3, 5 and 7 correspondingly — the first four prime numbers. In all recipes numbers should be two-digit. In the first recipe the second digit always is 0, in the second recipe the second digit always is 1, in the third recipe the first digit must be 0, in the fourth recipe the first digit always is 1. Consequently, the control numbers are as follows: in the first recipe $2 \times 5 + 11 = 21$ (the minimum prime divisor is 3), in the second recipe $3 \times 7 + 13 = 44$ (the minimum prime divisor is 2), in the third recipe $2 \times 3 + 17 = 23$ (the minimum prime divisor is 23) and, finally, in the fourth recipe $5 \times 7 + 19 = 54$ (the minimum prime divisor is 2).

In the third test, the number should consist of fourteen digits and be recorded in a sixteen-base numeral system. Number 0 (the number of the single bottles) matches, the control number will be equal to $10^{18} + 1$. The minimum prime divisor of this number is equal to 101 and you should print -1.