Problem Set 13: Strings 15-295 Spring 2018

A. War of the Corporations

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

A long time ago, in a galaxy far far away two giant IT-corporations Pineapple and Gogol continue their fierce competition. Crucial moment is just around the corner: Gogol is ready to release it's new tablet Lastus 3000.

This new device is equipped with specially designed artificial intelligence (AI). Employees of Pineapple did their best to postpone the release of Lastus 3000 as long as possible. Finally, they found out, that the name of the new artificial intelligence is similar to the name of the phone, that Pineapple released 200 years ago. As all rights on its name belong to Pineapple, they stand on changing the name of Gogol's artificial intelligence.

Pineapple insists, that the name of their phone occurs in the name of AI as a substring. Because the name of technology was already printed on all devices, the Gogol's director decided to replace some characters in AI name with "#". As this operation is pretty expensive, you should find the minimum number of characters to replace with "#", such that the name of AI doesn't contain the name of the phone as a substring.

Substring is a continuous subsequence of a string.

Input

The first line of the input contains the name of AI designed by Gogol, its length doesn't exceed $100\,000$ characters. Second line contains the name of the phone released by Pineapple 200 years ago, its length doesn't exceed 30. Both string are non-empty and consist of only small English letters.

Output

Print the minimum number of characters that must be replaced with "#" in order to obtain that the name of the phone doesn't occur in the name of AI as a substring.

Examples

input Copy

```
intellect
tell
```

output Copy

```
1
```

input Copy

```
google
apple
```

output Copy

```
0
```

input Copy

```
sirisiri
sir
```

output Copy

```
2
```

Note

In the first sample AI's name may be replaced with "`int#llect`".

In the second sample Gogol can just keep things as they are.

In the third sample one of the new possible names of AI may be "`s#ris#ri`".

B. Palindrome

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

Given a string s, determine if it contains any palindrome of length exactly 100 as a **subsequence**. If it has any, print any one of them. If it doesn't have any, print a palindrome that is a subsequence of s and is as long as possible.

Input

The only line of the input contains one string s of length n ($1 \le n \le 5 \cdot 10^4$) containing only lowercase English letters.

Output

If s contains a palindrome of length exactly 100 as a subsequence, print any palindrome of length 100 which is a subsequence of s. If s doesn't contain any palindromes of length exactly 100, print a palindrome that is a subsequence of s and is as long as possible.

If there exists multiple answers, you are allowed to print any of them.

Examples

input	Copy
bbbabcbbb	

output	Copy
bbbcbbb	

input	Copy
rquwmzexectvnbanemsmdufrg	

output	Copy
rumenanemur	

Note

A subsequence of a string is a string that can be derived from it by deleting some characters without changing the order of the remaining characters. A palindrome is a string that reads the same forward or backward.

C. Perfect Security

time limit per test: 3.5 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Alice has a very important message M consisting of some non-negative integers that she wants to keep secret from Eve. Alice knows that the only theoretically secure cipher is one-time pad. Alice generates a random key K of the length equal to the message's length. Alice computes the bitwise xor of each element of the message and the key ($A_i := M_i \oplus K_i$, where $\oplus$ denotes the bitwise XOR operation) and stores this encrypted message A. Alice is smart. Be like Alice. For example, Alice may have wanted to store a message $M = (0, 15, 9, 18)$. She generated a key $K = (16, 7, 6, 3)$. The encrypted message is thus $A = (16, 8, 15, 17)$. Alice realised that she cannot store the key with the encrypted message. Alice sent her key K to Bob and deleted her own copy. Alice is smart. Really, be like Alice.

Bob realised that the encrypted message is only secure as long as the key is secret. Bob thus randomly permuted the key before storing it. Bob thinks that this way, even if Eve gets both the encrypted message and the key, she will not be able to read the message. Bob is not smart. Don't be like Bob. In the above example, Bob may have, for instance, selected a permutation $(3, 4, 1, 2)$ and stored the permuted key $P = (6, 3, 16, 7)$. One year has passed and Alice wants to decrypt her message. Only now Bob has realised that this is impossible. As he has permuted the key randomly, the message is lost forever. Did we mention that Bob isn't smart? Bob wants to salvage at least some information from the message. Since he is not so smart, he asks for your help. You know the encrypted message A and the permuted key P. What is the lexicographically smallest message that could have resulted in the given encrypted text?

More precisely, for given A and P, find the lexicographically smallest message O, for which there exists a permutation π such that $O_i \oplus \pi(P_i) = A_i$ for every i.

Note that the sequence S is lexicographically smaller than the sequence T, if there is an index i such that $S_i < T_i$ and for all $j < i$ the condition $S_j = T_j$ holds.

Input

The first line contains a single integer N ($1 \le N \le 300000$), the length of the message.

The second line contains N integers $A_1, A_2, ..., A_N$ ($0 \le A_i < 2^{30}$) representing the encrypted message.

The third line contains N integers $P_1, P_2, ..., P_N$ ($0 \le P_i < 2^{30}$) representing the permuted encryption key.

Output

Output a single line with N integers, the lexicographically smallest possible message O. Note that all its elements should be non-negative.

Examples

input

```
3
8 4 13
17 2 7
```

output

```
10 3 28
```

input

```
5
12 7 87 22 11
18 39 9 12 16
```

output

```
0 14 69 6 44
```

input

```
10
331415699 278745619 998190004 423175621 42983144 166555524 843586353 802130100 337889448 685310951
226011312 266003835 342809544 504667531 529814910 684873393 817026985 844010788 993949858 1031395667
```

output

```
128965467 243912600 4281110 112029883 223689619 76924724 429589 119397893 613490433 362863284
```

Note

In the first case, the solution is $(10, 3, 28)$, since $8 \oplus 2 = 10, 4 \oplus 7 = 3$ and $13 \oplus 17 = 28$. Other possible permutations of key yield messages $(25, 6, 10)$, $(25, 3, 15)$, $(10, 21, 10)$, $(15, 21, 15)$ and $(15, 6, 28)$, which are all lexicographically larger than the solution.

D. Good Substrings

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You've got string s, consisting of small English letters. Some of the English letters are *good*, the rest are *bad*.

A substring $s[l...r]$ $(1 \le l \le r \le |s|)$ of string $s = s_1 s_2 ... s_{|s|}$ (where $|s|$ is the length of string s) is string $s_l s_{l+1} ... s_r$.

The substring $s[l...r]$ is *good*, if among the letters $s_l, s_{l+1}, ..., s_r$ there are **at most** k **bad** ones (look at the sample's explanation to understand it more clear).

Your task is to find the number of distinct good substrings of the given string s. Two substrings $s[x...y]$ and $s[p...q]$ are considered distinct if their content is different, i.e. $s[x...y] \neq s[p...q]$.

Input

The first line of the input is the non-empty string s, consisting of small English letters, the string's length is at most 1500 characters.

The second line of the input is the string of characters "0" and "1", the length is exactly 26 characters. If the i-th character of this string equals "1", then the i-th English letter is good, otherwise it's bad. That is, the first character of this string corresponds to letter "a", the second one corresponds to letter "b" and so on.

The third line of the input consists a single integer k $(0 \le k \le |s|)$ — the maximum acceptable number of bad characters in a good substring.

Output

Print a single integer — the number of distinct good substrings of string s.

Examples

input

```
ababab
01000000000000000000000000
1
```

output

```
5
```

input

```
acbacbacaa
00000000000000000000000000
2
```

output

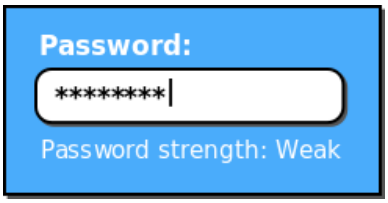
```
8
```

Note

In the first example there are following good substrings: "a", "ab", "b", "ba", "bab".

In the second example there are following good substrings: "a", "aa", "ac", "b", "ba", "c", "ca", "cb".

Passwords



It's that time of the year again when you go back to work and need to choose new passwords for all your services. The rules enforced by the system administrators are very strict, and the password you choose must obey the following restrictions:

- It must contain only letters and digits.
- It must have between A and B characters (inclusive).
- It must have at least one lowercase letter, one uppercase letter and one digit.
- It cannot contain any word of a collection of forbidden words (the blacklist).

A word of the blacklist is considered to be contained in the password if it appears as a substring, that is, if it occurs as a consecutive sequence of characters (regardless of it being upper or lower case). For instance, `swerc` is a substring of `SwErC`, `2016swerc2016` or `SWERC16`, but it is not a substring of `ICPC` or `sw16erc`.

Additionally, for the purposes of avoiding the blacklist, you cannot use *l33t*. Specifically, some digits can be interpreted as letters, namely the 0 ('o'), 1 ('i'), 3 ('e'), 5 ('s') and 7 ('t'). This implies that for example `5w3rC` would be an occurrence of `swerc`, and that `abcL337def` contains the word `leet`.

You cannot stop thinking about all these rules and you wonder how many different valid passwords there are... Can you calculate how many passwords would obey these rules?

Task

Given a blacklist with N words and two integers A and B, your task is to compute the number of different valid passwords that exist following the given constraints: made up of only letters and digits; length between A and B (inclusive); at least one lowercase letter, one uppercase letter, and one digit; no blacklisted substring. Since this number can be very big, compute it modulo 1 000 003.

Input

The first line contains two integers, A and B, specifying respectively the minimum and maximum length of the password. The second line contains an integer N, the number of words of the blacklist. The following N lines each contains a string W_i indicating a word in the blacklist. These words are formed only by lowercase letters.

Constraints

$3 \leq A \leq B \leq 20$	Size of the password
$0 \leq N \leq 50$	Number of words in the blacklist
$1 \leq \text{length}(W_i) \leq 20$	Size of each blacklist word

Output

The output should contain a single line with an integer indicating the number of valid passwords modulo 1 000 003. A valid password is one that respects all of the given constraints.

Sample Input

```
3 5
9
swerc
icpc
fbi
cia
bio
z
hi
no
yes
```

Sample Output

```
607886
```

Sample Explanation

In this case there are exactly 378 609 020 valid passwords and

$$378\,609\,020 \bmod 1\,000\,003 = 607\,886\,.$$

Some examples of valid passwords are: `aA1`, `B23tT`, `1g9K` or `B2j`.

Some examples of invalid passwords are: `aaA` (it does not contain digits), `12a` (it does not contain upper case letters), `a12A34` (length > 5) or `bB10` (contains `bio` as substring).

F — Insults

Insulting your friends and neighbors is the Ardenia national sport. However, as every sport, it has a certain set of rules which are quite hard to learn even by natives, not to mention the tourists. First of all, the insults are single words consisting entirely of four vowels: `a`, `e`, `i`, and `o`. But not all words that consist of these letters are insults. The only two letter insults are `ae` and `io`. If words w_1 and w_2 are insults, then words w_1w_2 and `a`w_1`e` and `i`w_1`o` are insults as well. Insults are created only in such way.

Usually, if somebody is insulting you, then you better have a sharp response insult prepared. Obviously, most of the answers are inappropriate. For example if you hear `aaeeio` (meaning *you fight like a dairy farmer*) you should not reply with `aeio` (*you are stupid*). To everybody's surprise, the linguists have found out that in all the cases the most appropriate reply is the insult of the same length and alphabetically next. Thus, for the insult above, the best reply would be `aaeioe` (*how appropriate; you fight like a cow*). This rule implies also the existence of so called ultimate insults, i.e., the ones for which there is no good reply. The eight letter ultimate insult is `ioioioio` (*your mother was a hamster and your father smelt of elderberries*).

Multiple Test Cases

The input contains several test cases. The first line of the input contains a positive integer $Z \leq 2000$, denoting the number of test cases. Then Z test cases follow, each conforming to the format described in section *Single Instance Input*. For each test case, your program has to write an output conforming to the format described in section *Single Instance Output*.

Single Instance Input

The input instance is one line containing a string of length at most 10^6. The only letters occurring in the string will be `a`, `e`, `i` and `o`.

Single Instance Output

You should output a single line containing the word `INVALID` if the string is not an insult. Otherwise, you should output the best reply to the insult read or word `ULTIMATE` if no such reply exists.

Example

Input	Output
3 eaeeio aaeeio ioioioioio	INVALID aaeioe ULTIMATE