E. Boolean Function

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

In this problem we consider Boolean functions of four variables A, B, C, D. Variables A, B, C and D are logical and can take values 0 or 1. We will define a function using the following grammar:

```
<expression> ::= <variable> | (<expression>) <operator> (<expression>)
```

```
<variable> ::= 'A' | 'B' | 'C' | 'D' | 'a' | 'b' | 'c' | 'd'
```

```
<operator> ::= '&' | '|'
```

Here large letters A, B, C, D represent variables, and small letters represent their negations. For example, if $A = 1$, then character `'A'` corresponds to value 1, and value character `'a'` corresponds to value 0. Here character '&' corresponds to the operation of logical AND, character '|' corresponds to the operation of logical OR.

You are given expression s, defining function f, where some operations and variables are missing. Also you know the values of the function $f(A, B, C, D)$ for some n distinct sets of variable values. Count the number of ways to restore the elements that are missing in the expression so that the resulting expression corresponded to the given information about function f in the given variable sets. As the value of the result can be rather large, print its remainder modulo $10^9 + 7$.

Input

The first line contains expression s ($1 \le |s| \le 500$), where some characters of the operators and/or variables are replaced by character '?'.

The second line contains number n ($0 \le n \le 2^4$) — the number of integers sets for which we know the value of function $f(A, B, C, D)$. Next n lines contain the descriptions of the sets: the i-th of them contains five integers a_i, b_i, c_i, d_i, e_i ($0 \le a_i, b_i, c_i, d_i, e_i \le 1$), separated by spaces and meaning that $f(a_i, b_i, c_i, d_i) = e_i$.

It is guaranteed that all the tuples (a_i, b_i, c_i, d_i) are distinct.

Output

In a single line print the answer to the problem.

Examples

input

```
?
2
1 0 1 0 1
0 1 1 0 1
```

output

```
2
```

input

```
(A)?(?)
1
1 1 0 0 0
```

output

```
4
```

input

```
((?)&(?))|((?)&(?))
0
```

output

```
4096
```

input

```
b
1
1 0 1 1 1
```

output

```
1
```

Note

In the first sample the two valid expressions are 'C' and 'd'.

In the second sample the expressions look as follows: '(A)&(a)', '(A)&(b)', '(A)&(C)', '(A)&(D)'.