Codeforces β
Sponsored by Telegram

Codeforces Round #402 (Div. 1)

A. String Game

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

Little Nastya has a hobby, she likes to remove some letters from word, to obtain another word. But it turns out to be pretty hard for her, because she is too young. Therefore, her brother Sergey always helps her.

Sergey gives Nastya the word t and wants to get the word p out of it. Nastya removes letters in a certain order (one after another, in this order strictly), which is specified by permutation of letters' indices of the word t: $a_1 \dots a_{|t|}$. We denote the length of word x as $|x|$. Note that after removing one letter, the indices of other letters don't change. For example, if $t =$ "nastya" and $a = [4, 1, 5, 3, 2, 6]$ then removals make the following sequence of words "nastya" "nas~~t~~ya" "~~n~~as~~t~~ya" "~~n~~as~~ty~~a" "~~n~~a~~sty~~a" "~~nasty~~a" "~~nastya~~".

Sergey knows this permutation. His goal is to stop his sister at some point and continue removing by himself to get the word p. Since Nastya likes this activity, Sergey wants to stop her as late as possible. Your task is to determine, how many letters Nastya can remove before she will be stopped by Sergey.

It is guaranteed that the word p can be obtained by removing the letters from word t.

Input

The first and second lines of the input contain the words t and p, respectively. Words are composed of lowercase letters of the Latin alphabet ($1 \le |p| < |t| \le 200\,000$). It is guaranteed that the word p can be obtained by removing the letters from word t.

Next line contains a permutation $a_1, a_2, \dots, a_{|t|}$ of letter indices that specifies the order in which Nastya removes letters of t ($1 \le a_i \le |t|$, all a_i are distinct).

Output

Print a single integer number, the maximum number of letters that Nastya can remove.

Examples

input

```
ababcba
abb
5 3 4 1 7 6 2
```

output

```
3
```

input

```
bbbabb
bb
1 6 3 4 2 5
```

output

```
4
```

Note

In the first sample test sequence of removing made by Nastya looks like this:

"ababcba" "abab~~c~~ba" "ab~~a~~b~~c~~ba" "ab~~abc~~ba"

Nastya can not continue, because it is impossible to get word "abb" from word "ab~~abc~~ba".

So, Nastya will remove only three letters.

B. Bitwise Formula

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Bob recently read about bitwise operations used in computers: `AND`, `OR` and `XOR`. He have studied their properties and invented a new game.

Initially, Bob chooses integer m, bit depth of the game, which means that all numbers in the game will consist of m bits. Then he asks Peter to choose some m-bit number. After that, Bob computes the values of n variables. Each variable is assigned either a constant m-bit number or result of bitwise operation. Operands of the operation may be either variables defined before, or the number, chosen by Peter. After that, Peter's score equals to the sum of all variable values.

Bob wants to know, what number Peter needs to choose to get the minimum possible score, and what number he needs to choose to get the maximum possible score. In both cases, if there are several ways to get the same score, find the minimum number, which he can choose.

Input

The first line contains two integers n and m, the number of variables and bit depth, respectively ($1 \le n \le 5000$; $1 \le m \le 1000$).

The following n lines contain descriptions of the variables. Each line describes exactly one variable. Description has the following format: name of a new variable, space, sign "`:=`", space, followed by one of:

1. Binary number of exactly m bits.
2. The first operand, space, bitwise operation ("`AND`", "`OR`" or "`XOR`"), space, the second operand. Each operand is either the name of variable defined before or symbol '`?`', indicating the number chosen by Peter.

Variable names are strings consisting of lowercase Latin letters with length at most 10. All variable names are different.

Output

In the first line output the minimum number that should be chosen by Peter, to make the sum of all variable values minimum possible, in the second line output the minimum number that should be chosen by Peter, to make the sum of all variable values maximum possible. Both numbers should be printed as m-bit binary numbers.

Examples

input

```
3 3
a := 101
b := 011
c := ? XOR b
```

output

```
011
100
```

input

```
5 1
a := 1
bb := 0
cx := ? OR a
d := ? XOR ?
e := d AND bb
```

output

```
0
0
```

Note

In the first sample if Peter chooses a number 011_2, then $a = 101_2, b = 011_2, c = 000_2$, the sum of their values is 8. If he chooses the number 100_2, then $a = 101_2, b = 011_2, c = 111_2$, the sum of their values is 15.

For the second test, the minimum and maximum sum of variables a, bb, cx, d and e is 2, and this sum doesn't depend on the number chosen by Peter, so the minimum Peter can choose is 0.

C. Peterson Polyglot

time limit per test: 4 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Peterson loves to learn new languages, but his favorite hobby is making new ones. Language is a set of words, and word is a sequence of lowercase Latin letters.

Peterson makes new language every morning. It is difficult task to store the whole language, so Peterson have invented new data structure for storing his languages which is called *broom*. Broom is rooted tree with edges marked with letters. Initially broom is represented by the only vertex — the root of the broom. When Peterson wants to add new word to the language he stands at the root and processes the letters of new word one by one. Consider that Peterson stands at the vertex u. If there is an edge from u marked with current letter, Peterson goes through this edge. Otherwise Peterson adds new edge from u to the new vertex v, marks it with the current letter and goes through the new edge. Size of broom is the number of vertices in it.

In the evening after working day Peterson can't understand the language he made this morning. It is too difficult for bored Peterson and he tries to make it simpler. Simplification of the language is the process of erasing some letters from some words of this language. Formally, Peterson takes some positive integer p and erases p-th letter from all the words of this language having length at least p. Letters in words are indexed starting by 1. Peterson considers that simplification should change at least one word, i.e. there has to be at least one word of length at least p. Peterson tries to make his language as simple as possible, so he wants to choose p such that the size of the broom for his simplified language is as small as possible.

Peterson is pretty annoyed with this task so he asks you for help. Write a program to find the smallest possible size of the broom and integer p.

Input

The first line of input contains integer n $(2 \le n \le 3 \cdot 10^5)$ — the size of the broom.

Next $n - 1$ lines describe the broom: i-th of them contains integers u_i, v_i and letter x_i — describing the edge from u_i to v_i marked with letter x_i.

Vertices are numbered from 1 to n. All x_i are lowercase latin letters. Vertex 1 is the root of the broom.

Edges describe correct broom which is made from Peterson's language.

Output

The first line of output should contain the minimum possible size of the broom after its simplification. The second line of output should contain integer p to choose. If there are several suitable p values, print the smallest one.

Examples

input

```
5
1 2 c
2 3 a
3 4 t
2 5 t
```

output

```
3
2
```

input

```
16
1 2 o
2 3 f
1 4 p
4 5 i
5 6 e
6 7 c
7 8 e
4 9 r
9 10 e
10 11 t
11 12 t
12 13 y
10 14 f
14 15 i
15 16 x
```

output

```
12
2
```

Note

Broom from the second sample test can be built using language "piece", "of", "pie", "pretty", "prefix". Its simplification with $p = 2$ obtains the language of words "pece", "o", "pe", "petty", "pefix". This language gives us the broom with minimum possible size.

D. Parquet Re-laying

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

Peter decided to lay a parquet in the room of size $n \times m$, the parquet consists of tiles of size 1×2. When the workers laid the parquet, it became clear that the tiles pattern looks not like Peter likes, and workers will have to re-lay it.

The workers decided that removing entire parquet and then laying it again is very difficult task, so they decided to make such an operation every hour: remove two tiles, which form a 2×2 square, rotate them 90 degrees and put them back on the same place.

They have no idea how to obtain the desired configuration using these operations, and whether it is possible at all.

Help Peter to make a plan for the workers or tell that it is impossible. The plan should contain at most $100\,000$ commands.

Input

The first line contains integer n and m, size of the room ($1 \le n, m \le 50$). At least one of them is even number.

The following n lines contain m characters each, the description of the current configuration of the parquet tiles. Each character represents the position of the half-tile. Characters '`L`', '`R`', '`U`' and '`D`' correspond to the left, right, upper and lower halves, respectively.

The following n lines contain m characters each, describing the desired configuration in the same format.

Output

In the first line output integer k, the number of operations. In the next k lines output description of operations. The operation is specified by coordinates (row and column) of the left upper half-tile on which the operation is performed.

If there is no solution, output `-1` in the first line.

Examples

input

```
2 3
ULR
DLR
LRU
LRD
```

output

```
2
1 2
1 1
```

input

```
4 3
ULR
DLR
LRU
LRD
ULR
DUU
UDD
DLR
```

output

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

Note

In the first sample test first operation is to rotate two rightmost tiles, after this all tiles lie vertically. Second operation is to rotate two leftmost tiles, after this we will get desired configuration.

E. Selling Numbers

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

Boris really likes numbers and even owns a small shop selling interesting numbers. He has n decimal numbers B_i. Cost of the number in his shop is equal to the sum of costs of its digits. You are given the values c_d, where c_d is the cost of the digit d. Of course, Boris is interested in that numbers he owns have the maximum cost possible.

Recently Boris got hold of the magical artifact A, which can allow him to increase the cost of his collection. Artifact is a string, consisting of digits and '`?`' symbols. To use the artifact, Boris must replace all '`?`' with digits to get a decimal number without leading zeros (it is also not allowed to get number 0). After that, the resulting number is added to all numbers B_i in Boris' collection. He uses the artifact exactly once.

What is the maximum cost of the collection Boris can achieve after using the artifact?

Input

First line contains artifact A, consisting of digits '`0`'–'`9`' and '`?`' symbols ($1 \le |A| \le 1000$). Next line contains n — the amount of numbers in Boris' collection ($1 \le n \le 1000$). Next n lines contain integers B_i ($1 \le B_i < 10^{1000}$). A doesn't start with '`0`'.

Last line contains ten integers — costs of digits $c_0, c_1, ..., c_9$ ($0 \le c_i \le 1000$).

Output

Output one integer — the maximum possible cost of the collection after using the artifact.

Examples

input

```
42
3
89
1
958
0 0 1 1 2 2 3 3 4 4
```

output

```
4
```

input

```
?5?
4
2203
5229
276
6243
2 1 6 1 1 2 5 2 2 3
```

output

```
62
```

Note

In the second sample input, the optimal way is to compose the number `453`. After adding this number, Boris will have numbers `2656`, `5682`, `729` and `6696`. The total cost of all digits in them is equal to $18 + 15 + 11 + 18 = 62$.