Croc Champ 2012 - Round 2

A. Trading Business

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

To get money for a new aeonic blaster, ranger Qwerty decided to engage in trade for a while. He wants to buy some number of items (or probably not to buy anything at all) on one of the planets, and then sell the bought items on another planet. Note that this operation is not repeated, that is, the buying and the selling are made only once. To carry out his plan, Qwerty is going to take a bank loan that covers all expenses and to return the loaned money at the end of the operation (the money is returned without the interest). At the same time, Querty wants to get as much profit as possible.

The system has n planets in total. On each of them Qwerty can buy or sell items of m types (such as food, medicine, weapons, alcohol, and so on). For each planet i and each type of items j Qwerty knows the following:

- a_{ij} — the cost of buying an item;
- b_{ij} — the cost of selling an item;
- c_{ij} — the number of remaining items.

It is not allowed to buy more than c_{ij} items of type j on planet i, but it is allowed to sell any number of items of any kind.

Knowing that the hold of Qwerty's ship has room for no more than k items, determine the maximum profit which Qwerty can get.

Input

The first line contains three space-separated integers n, m and k ($2 \le n \le 10$, $1 \le m, k \le 100$) — the number of planets, the number of question types and the capacity of Qwerty's ship hold, correspondingly.

Then follow n blocks describing each planet.

The first line of the i-th block has the planet's name as a string with length from 1 to 10 Latin letters. The first letter of the name is uppercase, the rest are lowercase. Then in the i-th block follow m lines, the j-th of them contains three integers a_{ij}, b_{ij} and c_{ij} ($1 \le b_{ij} < a_{ij} \le 1000$, $0 \le c_{ij} \le 100$) — the numbers that describe money operations with the j-th item on the i-th planet. The numbers in the lines are separated by spaces.

It is guaranteed that the names of all planets are different.

Output

Print a single number — the maximum profit Qwerty can get.

Sample test(s)

input
3 3 10 Venus 6 5 3 7 6 5 8 6 10 Earth 10 9 0 8 6 4 10 9 3 Mars 4 3 0 8 4 12 7 2 5
output
16

Note

In the first test case you should fly to planet `Venus`, take a loan on 74 units of money and buy three items of the first type and 7 items of the third type ($3 \cdot 6 + 7 \cdot 8 = 74$). Then the ranger should fly to planet `Earth` and sell there all the items he has bought. He gets $3 \cdot 9 + 7 \cdot 9 = 90$ units of money for the items, he should give 74 of them for the loan. The resulting profit equals 16 units of money. We cannot get more profit in this case.

B. Word Cut

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

Let's consider one interesting word game. In this game you should transform one word into another through special operations.

Let's say we have word w, let's split this word into two non-empty parts x and y so, that $w = xy$. A *split* operation is transforming word $w = xy$ into word $u = yx$. For example, a *split* operation can transform word "wordcut" into word "cutword".

You are given two words $start$ and end. Count in how many ways we can transform word $start$ into word end, if we apply exactly k *split* operations consecutively to word $start$.

Two ways are considered different if the sequences of applied operations differ. Two operation sequences are different if exists such number i ($1 \le i \le k$), that in the i-th operation of the first sequence the word splits into parts x and y, in the i-th operation of the second sequence the word splits into parts a and b, and additionally $x \neq a$ holds.

Input

The first line contains a non-empty word $start$, the second line contains a non-empty word end. The words consist of lowercase Latin letters. The number of letters in word $start$ equals the number of letters in word end and is at least 2 and doesn't exceed 1000 letters.

The third line contains integer k ($0 \le k \le 10^5$) — the required number of operations.

Output

Print a single number — the answer to the problem. As this number can be rather large, print it modulo 1000000007 ($10^9 + 7$).

Sample test(s)

input
ab ab 2
output
1

input
ababab ababab 1
output
2

input
ab ba 2
output
0

Note

The sought way in the first sample is:

ab → a|b → ba → b|a → ab

In the second sample the two sought ways are:

- ababab → abab|ab → ababab
- ababab → ab|abab → ababab

C. Playing with Superglue

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

Two players play a game. The game is played on a rectangular board with $n \times m$ squares. At the beginning of the game two different squares of the board have two chips. The first player's goal is to shift the chips to the same square. The second player aims to stop the first one with a *tube of superglue*.

We'll describe the rules of the game in more detail.

The players move in turns. The first player begins.

With every move the first player chooses one of his unglued chips, and shifts it one square to the left, to the right, up or down. It is not allowed to move a chip beyond the board edge. At the beginning of a turn some squares of the board may be covered with a glue. The first player can move the chip to such square, in this case the chip gets tightly glued and cannot move any longer.

At each move the second player selects one of the free squares (which do not contain a chip or a glue) and covers it with superglue. The glue dries long and squares covered with it remain sticky up to the end of the game.

If, after some move of the first player both chips are in the same square, then the first player wins. If the first player cannot make a move (both of his chips are glued), then the second player wins. Note that the situation where the second player cannot make a move is impossible — he can always spread the glue on the square from which the first player has just moved the chip.

We will further clarify the case where both chips are glued and are in the same square. In this case the first player wins as the game ends as soon as both chips are in the same square, and the condition of the loss (the inability to move) does not arise.

You know the board sizes and the positions of the two chips on it. At the beginning of the game all board squares are glue-free. Find out who wins if the players play optimally.

Input

The first line contains six integers n, m, x_1, y_1, x_2, y_2 — the board sizes and the coordinates of the first and second chips, correspondingly ($1 \le n, m \le 100; 2 \le n \times m; 1 \le x_1, x_2 \le n; 1 \le y_1, y_2 \le m$). The numbers in the line are separated by single spaces.

It is guaranteed that the chips are located in different squares.

Output

If the first player wins, print "`First`" without the quotes. Otherwise, print "`Second`" without the quotes.

Sample test(s)

input
1 6 1 2 1 6
output
First

input
6 5 4 3 2 1
output
First

input
10 10 1 1 10 10
output
Second

D. Hyper String

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

Paul Erdős's prediction came true. Finally an alien force landed on the Earth. In contrary to our expectation they didn't asked the humans to compute the value of a Ramsey number (maybe they had solved it themselves). They asked another question which seemed as hard as calculating Ramsey numbers. Aliens threatened that if humans don't solve this problem in less than 2 hours they will destroy the Earth.

Before telling the problem they introduced the concept of Hyper Strings. A Hyper String is made by concatenation of some base strings. Suppose you are given a list of base strings $b_1, b_2, ..., b_n$. Now the Hyper String made from indices list $i_1, i_2, ..., i_m$ is concatenation of base strings $b_{i_1}, b_{i_2}, ..., b_{i_m}$. A Hyper String can be very large and doing operations on it is very costly for computers.

The aliens asked humans to compute the length of the longest common sub-sequence of a Hyper String t with a string s.

Input

The first line of input contains the single integer n ($1 \le n \le 2000$) — the number of base strings.

The next n lines contains values of base strings. Each base string is made of lowercase Latin letters. A base string cannot be empty string and the sum of lengths of all n base strings doesn't exceed 10^6.

The next line contains the single integer m ($1 \le m \le 2000$) — the number of base strings in the given Hyper String t.

The next line contains m space-separated integer numbers $i_1, i_2, ..., i_m$ ($1 \le i_j \le n$) — the indices of base strings in the Hyper String t.

The last line contains a non-empty string s. String s is made of lowercase Latin letters and its length is no more than 2000 characters.

Output

Print the length of longest common sub-sequence of Hyper String t and string s. If there is no common sub-sequence print 0.

Sample test(s)

input
2 cba dgh 2 1 2 aedfhr
output
3

input
2 b a 5 1 2 1 2 1 aaa
output
2

Note

The *length* of string s is the number of characters in it. If the length of string s is marked as $|s|$, then string s can be represented as $s = s_1 s_2 ... s_{|s|}$.

A non-empty string $y = s[p_1 p_2 ... p_{|y|}] = s_{p_1} s_{p_2} ... s_{p_{|y|}}$ ($1 \le p_1 < p_2 < ... < p_{|y|} \le |s|$) is a *subsequence* of string s. For example, "coders" is a subsequence of "codeforces".

E. Archaeology

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

This time you should help a team of researchers on an island in the Pacific Ocean. They research the culture of the ancient tribes that used to inhabit the island many years ago.

Overall they've dug out n villages. Some pairs of villages were connected by roads. People could go on the roads in both directions. Overall there were exactly $n - 1$ roads, and from any village one could get to any other one.

The tribes were not peaceful and they had many wars. As a result of the wars, some villages were destroyed completely. During more peaceful years some of the villages were restored.

At each moment of time people *used* only those roads that belonged to some shortest way between two villages *that existed at the given moment*. In other words, people used the minimum subset of roads in such a way, that it was possible to get from any existing village to any other existing one. Note that throughout the island's whole history, there existed exactly $n - 1$ roads that have been found by the researchers. There never were any other roads.

The researchers think that observing the total sum of used roads' lengths at different moments of time can help to better understand the tribes' culture and answer several historical questions.

You will be given the full history of the tribes' existence. Your task is to determine the total length of used roads at some moments of time.

Input

The first line contains an integer n ($1 \le n \le 10^5$) — the number of villages. The next $n - 1$ lines describe the roads. The i-th of these lines contains three integers a_i, b_i and c_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$, $1 \le c_i \le 10^9$, $1 \le i < n$) — the numbers of villages that are connected by the i-th road and the road's length. The numbers in the lines are separated by a space.

The next line contains an integer q ($1 \le q \le 10^5$) — the number of queries. Then follow q queries, one per line, ordered by time. Each query belongs to one of three types:

- "`+ x`" — village number x is restored ($1 \le x \le n$).
- "`- x`" — village number x is destroyed ($1 \le x \le n$).
- "`?`" — the archaeologists want to know the total length of the roads which were used for that time period.

It is guaranteed that the queries do not contradict each other, that is, there won't be queries to destroy non-existing villages or restore the already existing ones. It is guaranteed that we have at least one query of type "`?`". It is also guaranteed that one can get from any village to any other one by the given roads.

At the initial moment of time no village is considered to exist.

Output

For each query of type "`?`" print the total length of used roads on a single line. You should print the answers to the queries in the order, in which they are given in the input.

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

Sample test(s)

input

```
6
1 2 1
1 3 5
4 1 7
4 5 3
6 4 2
10
+ 3
+ 1
?
+ 6
?
+ 5
?
- 6
- 3
?
```

output

```
5
14
17
10
```