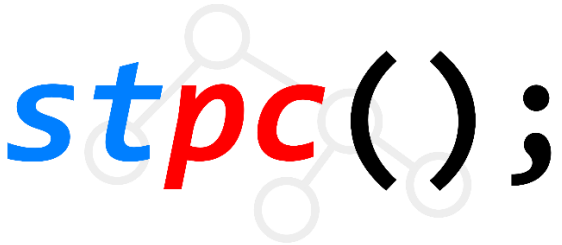


Buddhist Sin Tak College Competitive Programming Team Mid-Term Examination (Senior Group)

Task Overview

ID	Name	Time Limit	Memory Limit	Subtasks
A	Decode III	2.000 s	256 MB	7 + 16 + 20 + 23 + 34
B	Yet Another String Algorithm	1.000 s	256 MB	10 checkpoints
C	Number Snake	1.000 s	256 MB	14 + 11 + 27 + 48
D	Mayan Writing	1.000 s	256 MB	16 checkpoints

Notice:

Unless otherwise specified, inputs and outputs shall follow the format below:

- One space between a number and another number or character in the same line.
- No space between characters in the same line.
- Each string shall be placed in its own separate line.
- Outputs will be automatically fixed as follows: Trailing spaces in each line will be removed and an end-of-line character will be added to the end of the output if not present. All other format errors will not be fixed.

Only **C++** is allowed to be used in this examination.

C++ programmers should be aware that using C++ streams (`cin` / `cout`) may lead to I/O bottlenecks and substantially lower performance.

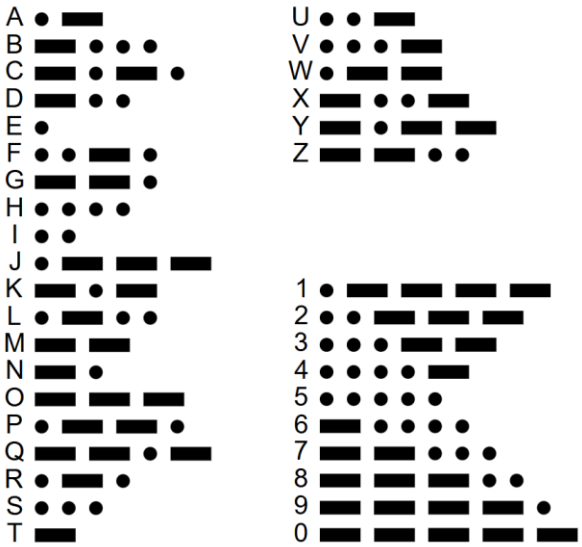
For some problems 64-bit integers may be required. In C++ it is `long long` and its token for `scanf` or `prinf` is `%lld`.

All tasks are divided into subtasks. You need to pass all test cases in a subtask to get points.

A Decode III

Time Limit: 2.000 s Memory Limit: 256 MB

Byteland received secret signals from their undercover in Bitland. Due to their convenience, the undercover used a telecommunication method called Morse Code to encode text characters as standardized sequences of two different characters, called *dots* (.) and *dashes* (-). Here are the international morse code conversion table from Wikipedia:



For example, the word “HELLO” will be encrypted as “.... . .-.. .-.. ---”. Unfortunately, the table is widely known, including Bitland. Therefore, the undercover transmitted the signal without any whitespaces. The actual signal received by Byteland if the word “HELLO” is sent will be “.....-...-..---”.

Byteland would like to decode the message by inserting whitespace(s) within the Morse code signal. Obviously, there are several rules for Byteland to follow in order to decode the message correctly:

- Whitespace(s) could not be inserted at the beginning or at the end of the message.
- Any number of whitespace(s) can be inserted, including no whitespaces are inserted.
- The message must be fully decoded by the conversion table stated above.

At this time, Byteland received another encrypted message with N characters. A question arises: how many distinct messages can be decoded by inserting whitespace(s) within the message? As the number may be very large, **output the number modulo** $10^9 + 7$.

INPUT

Input two lines.

The first line contains an integer N.

The second line contains a string with length N consists of *dots* (.) and *dashes* (-), which is the encrypted message received by Byteland.

OUTPUT

Output an integer, which is the number of distinct messages can be decoded by inserting whitespace(s) within the message modulo $10^9 + 7$.

SAMPLES

	Input	Output
1	4 ---.	7

The message can be decoded as “OE”, “MN”, “MTE”, “TG”, “TME”, “TTN”, “TTTE”.

	Input	Output
2	16 .....-...-..---	22159

SUBTASKS

For all test cases,

$1 \le N \le 2 \times 10^6$

Denote s be the encrypted message. s_i means the i-th character of the message, where $1 \le i \le N$.

It is guaranteed that the encrypted message can be decoded in some way.

Points are given per **subtask** in this problem. You have to pass all the checkpoint in the subtask in order to get the points of the subtask.

	Score	Constraints
1	7	$N \le 20$ s_i is dot for $1 \le i \le N$
2	16	$2 \le N \le 20$ s_i is dot for $1 \le i \le N-2$ s_i is dash for $N-1 \le i \le N$
3	20	$N \le 20$
4	23	$N \le 3000$
5	34	No additional constraints

B Yet Another String Algorithm

Time Limit: 1.000 s Memory Limit: 256 MB

Alice is studying string algorithms. Let S and T be two strings with index in 0-based. Alice wants to find the index of the beginning of the first occurrence of S in T. The pseudocode of her algorithm is:

```
subprogram match(S, T)
    N ← size of S
    M ← size of T
    for i from 0 to N - M
        j ← 0
        while j < M do
            if S[i + j] != T[j] then
                break
            j ← j + 1
        if j = M then
            return i
    return -1           // Not found
```

However, the time complexity for her algorithm is $O(NM)$, which may be a bad idea for her. Recently, she learnt a new algorithm called the Knuth-Morris-Pratt algorithm (or KMP algorithm) to tackle this problem within $O(N+M)$.

The core idea of KMP algorithm is based on **prefix function** array P with size M of T with size M. $P[i]$ means the maximum length of proper prefix which is equal to its proper suffix in $S[1..i]$. Note that the string itself is neither its proper prefix nor proper suffix. P can be calculated as follow:

$$P[i]=\max_{k\in S}\left(P[k-1]+1 \text{ if } T[i]=T\left[P[k-1]\right],0\right) \text{ where } S=\left\{i,P[i-1],P\left[P[i-1]-1\right],\cdots\right\}$$

After computing P, the KMP algorithm can be implemented as follows:

```
subprogram KMP(S, T, P)
    N ← size of S
    M ← size of T
    i ← 0
    j ← 0
    while i < N and j < M do
        if S[i] = T[j] then
            i ← i + 1
            j ← j + 1
        else if j = 0 then
            i ← i + 1
        else then
            j ← P[j-1]
    if j = M then
        return i - j
    return -1           // Not found
```

Alice finishes her implementation quickly. Now, it is your job to implement the KMP algorithm as a classmate of Alice.

INPUT

Input three lines.

The first line contains two integers N and M, which is the length of string S and T respectively.

The second line contains a string S.

The third line contains a string T.

It is guaranteed that all characters in both string can be typed using a keyboard and visible.

OUTPUT

Output two lines.

The first line contains M integers, which is the content of P. The i-th integer will be equal to $P[i]$.

The second line contains an integers, which is the index of the beginning of the first occurrence of S in T. If S does not exist in T, output -1.

SAMPLES

	Input	Output
1	9 3 ABCABCABB CAB	0 0 0 2
2	26 9 ABCDEFGHIJKLMNOPQRSTUVWXYZ ABCABCABB	0 0 0 1 2 3 4 5 0 -1

SUBTASKS

For all test cases,

$1 \le N \le 10^7$

$1 \le M \le 10^7$

Points are given per **checkpoint** in this problem. You can get the point of the checkpoint when you pass them.

There are 10 checkpoints in total, each carry 10 points.

C Number Snake

Time Limit: 1.000 s Memory Limit: 256 MB

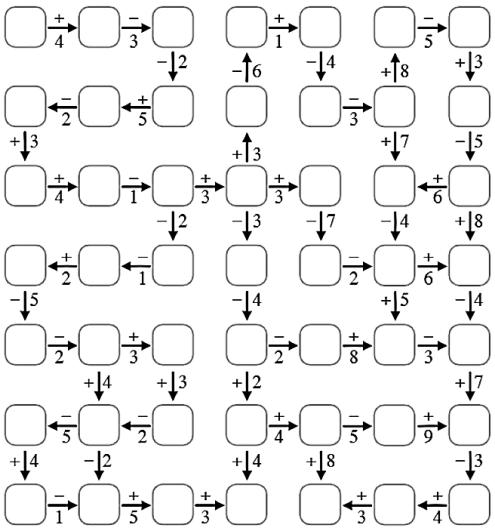
Alice is playing a game called Number Snake, which is a common Mathematics puzzle.

Number Snake consists of N squares, numbered from 1 to N. Alice will fill an integer A_i on the i-th square. Moreover, Alice should follow M rules witht either of the formats:

- $A_i + k = A_j$
- $A_i - k = A_j$

where k is an integer.

Alice would like to fill in all squares with integers between X and Y **inclusive**. Could you help her fill in all squares if possible?



An example of Number Snake

SCORING

For each test case, if you output `Possible` / `Impossible` with a valid construction, you score 100% on the test case.

Otherwise, if you output `Possible` / `Impossible` with *any* construction, you score 40% on the test case. Otherwise, you score 0% on the test case.

The points you get on a subtask is the minimum points you score across all test cases in the subtask.

If your output violates the format specified above, your score will be 0.

INPUT

The first line contains four integers, N, M, X and Y.

Each of the following M lines contains four integers x, y, k and c, which indicates a single rule:

- If $c = 1$, then the rule is $A_x + k = A_y$.
- If $c = 0$, then the rule is $A_x - k = A_y$.

OUTPUT

If Alice could not fill in all squares with integers between X and Y inclusive, output `Impossible`.

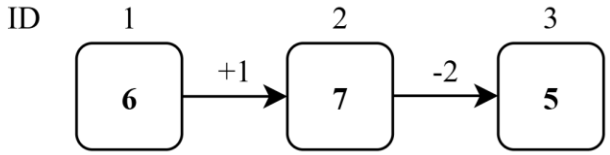
Otherwise, output `Possible` in the first line.

Then, for the following i-th line, output a single integer A_i.

SAMPLES

1

Input	Output
3 2 5 8 1 2 1 1 2 3 2 0	Possible 6 7 5



There exists an other construction: $A_1 = 7, A_2 = 8, A_3 = 6$.

2

5 4 -100 100 1 2 1 1 2 3 2 1 3 4 3 1 4 1 4 1	Impossible

3

5 3 1 4 1 2 3 1 2 3 2 0 4 5 5 1	Impossible

SUBTASKS

For all test cases,

$1 \le N \le 10^4$

$0 \le M \le \frac{N(N+1)}{2}$

$-10^5 \le X \le Y \le 10^5$

$-10^5 \le k \le 10^5$

There is at most one rule between two squares.

Points are given per **subtask** in this problem. You have to pass all the checkpoint in the subtask in order to get the points of the subtask.

	Score	Constraints
1	14	$M = N - 1$ For the i-th rule, $x = i, y = i + 1$
2	11	$M = N$ For the i-th rule, $x = i, y = (i \bmod N) + 1$
3	27	$M = N - 1$ For the i-th rule, $x = i, y = i + 1$ or $x = i + 1, y = i$
4	48	No additional constraints

D Mayan Writing

Time Limit: 1.000 s Memory Limit: 256 MB

Deciphering the Mayan writing has proven to be a harder task than anticipated by the early investigations. Mayan writing is based on small drawings known as glyphs which represent sounds. Mayan words are normally written as glyphs put together at various positions. Here are some examples of glyphs:

One of several problems in deciphering Mayan writing arises in the order of reading. When placing several glyphs in order to form a word, Mayan writers sometimes decided the position based more on their own esthetic views than on any particular rule. This leads to the fact that, even though the sound for many glyphs is known, sometimes archaeologists are not sure how to pronounce a written word.

The archaeologists are looking for a special word W. They know the glyphs for it, but they don't know all the possible ways of arranging them. They will provide you with the N glyphs from W and a sequence S with length M of all the glyphs (in the order they appear) in the carvings they are studying. They want you to count the number of possible appearances of the word W, which a possible appearance is a group of consecutive N glyphs in S that is a permutation of the glyphs in W.

INPUT

Input three lines.

The first line contains two integers, N and M.

The second line contains a string W, consists of uppercase letters and lowercase letters.

The third line contains a string S, consists of uppercase letters and lowercase letters.

OUTPUT

Output an integer, which is the number of possible appearances of W in S.

SAMPLES

	Input	Output
1	4 11 cAda AbrAcadAbRa	2

There are two possible appearances of W in S: “Acad” and “cadA”.

SUBTASKS

For all test cases,

$1 \le N \le 3000$

$1 \le M \le 3 \times 10^6$

Points are given per **checkpoint** in this problem. You can get the point of the checkpoint when you pass them.

There are 16 checkpoints in total, each carry around 6.3 points evenly.

For 50% of cases, $N \le 10$.