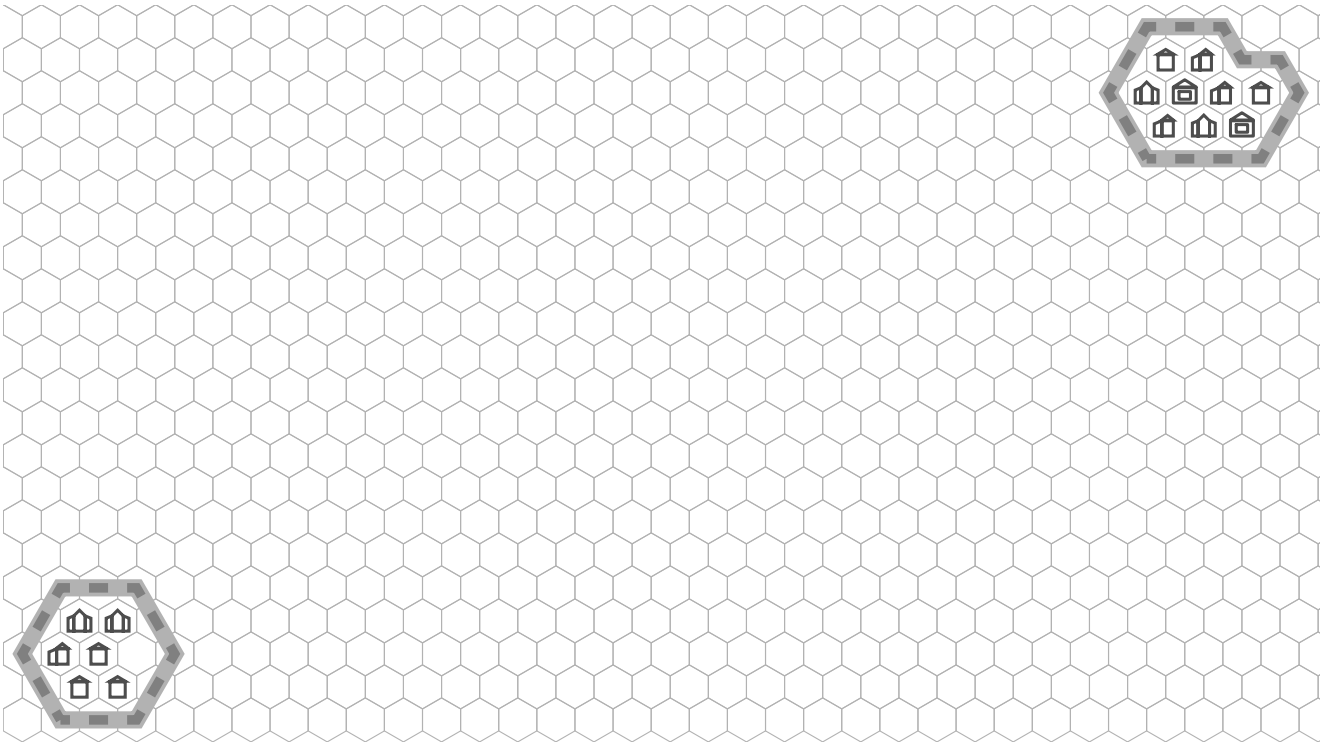
Problem A. City Wall *(Division 2)*

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

One of Petya's exercise books is amazing. On each sheet, the print is not some squares or lines, but a hexagonal grid, just like in his favorite computer games. A piece of his excercise book is shown below.



Petya wants to draw a city in his exercise book. The city will consist of n houses, each house will occupy one hex of the grid, and different houses will occupy different hexes. As n in this problem can be large, below we will pretend the hexagonal sheet is infinite in all directions.

To protect the city from barbarians, Petya has to build a thick wall around it. Formally, the wall will occupy several hexes of the grid. The length of the wall d is the number of hexes in it. Furthermore, if a guard patrols the wall starting from some hex and going around the city by moving between adjacent hexes, then after d moves the guard must visit each hex of the wall exactly once and return to the initial hex. Surely, all houses of the city must stand in the part of the sheet which is surrounded by the wall.

Petya realizes that building a wall takes time and resources. So he wants to pick hexes for the houses and for the wall in such a way that the length of the wall d is as small as possible. What is the least possible value of d?

Input

The first line contains a single integer n: the number of houses in the city ($1 \le n \le 10^9$).

Output

On the first line, print a single integer d: the minimum possible length of the wall.

Examples

standard input	standard output
6	12
9	14

Problem B. Domino Colorings *(Division 2)*

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Alice wants to become an artist. To do that she needs a lot of practice. But she does not know what to draw! She asked her friend Bob for help and he came up with an idea.

Bob likes to play with dominoes. His dominoes are colored: one half is black and other is white. He decided to tile an $n \times m$ rectangular grid with dominoes and let Alice draw it.

The picture will look like $n \times m$ grid with each cell colored either black or white, but the domino borders will not be drawn. Bob can tile the grid in many ways, so Alice can draw a lot of pictures! But sometimes different tilings can produce the same picture.

Bob wants to give Alice as much practice as he can, but he does not want Alice to waste her time drawing the same picture twice. Help him count the number of different pictures that Alice can draw. As the answer can be very large, output it modulo $10^9 + 7$.

Input

The first line contains integers n and m: the dimensions of the grid ($1 \le n \le 5$, $1 \le m \le 5$).

Output

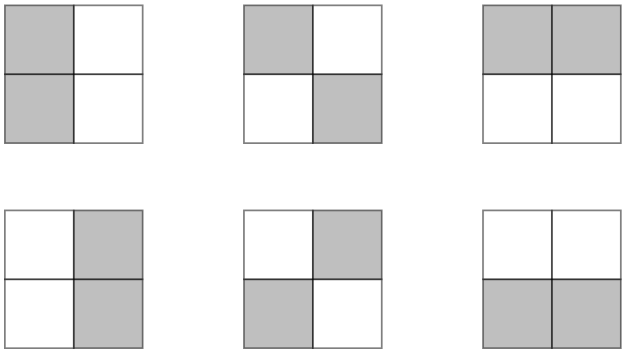
Print a single integer: the number of different pictures modulo $10^9 + 7$.

Examples

standard input	standard output
2 2	6
2 3	16
3 3	0

Explanations

In the first example, there are six pictures:



Notice that the second picture could be produced from two tilings (borders of dominoes are in bold):

Problem C. Counterquestion *(Division 2)*

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

In the year 1988, in IX issue of the "Kvant" magazine, the following problem was published:

Find the numeric value of the word ТРАНСПОРТИРОВКА*, where the same letters are replaced by the same digits correpondingly, and different letters are replaced by different digits, in such a way that all inequalities on the picture are true.*

(The picture for this problem shows the chain of inequalities Т>Р>А>Н<С<П<О<Р<Т>И>Р>О<В<К<А.)

Afterwards, this problem was widely used at various mathematical tourneys and olympiads for juniors. Recently, it can be encountered at such events each year.

Today you are a jury member at one of these olympiads. Like other problemsetters, you want to offer this problem to your contestants. But you are worried that some of them might already know the answer by heart, so the jury has to change the statement slightly. Specifically, it is necessary to replace the word ТРАНСПОРТИРОВКА by some other word, and place comparison signs in that new word (signs of inequality or equality in case of double letters). The word is chosen by another member of the jury. You know that she can choose any word from a dictionary with the restriction that the word must contain exactly ten different letters. The dictionary she uses is described in the Input section. As for your part, you have to place comparison signs in the given word, or determine that it is impossible to place the signs in such a way that the problem has a unique answer.

Input

The first line of input contains a word from the dictionary. The word consists of small English letters and contains exactly ten different letters. The dictionary used for this problem is the public domain word list ENABLE for word games in English. The version of dictionary used in this problem can be downloaded here: `http://acm.math.spbu.ru/171217/words.unix.txt` with Unix line endings or `http://acm.math.spbu.ru/171217/words.windows.txt` with Windows line endings.

Output

On the first line of output, print a chain of inequalities that has a unique answer (numeric value of the given word). The chain must contain letters of a given word in their original order, separated by comparison signs (">", "<", "="). All the letters and signs must be separated by spaces. In case there exist multiple suitable chains, output any one of them. If no suitable chain exists for a given word, output "`Impossible`".

Examples

standard input	standard output
counterquestion	c > o > u < n > t < e < r < q < u > e > s > t > i < o > n
nostalgically	n > o > s > t > a > l < g < i < c < a > l = l > y
hyperparasitism	Impossible

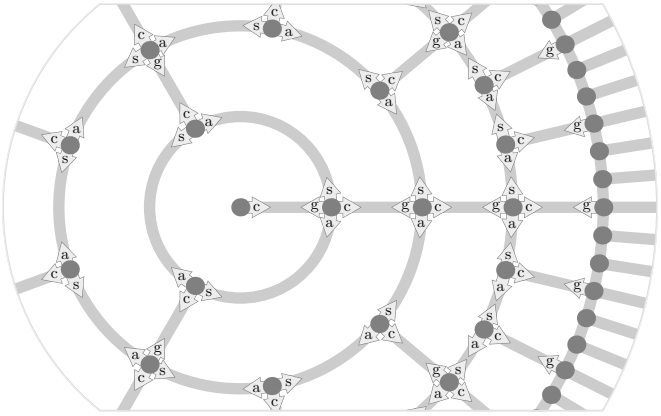
Explanations

Here is the chain given as the answer for the first test and the unique solution for this chain:
c > o > u < n > t < e < r < q < u > e > s > t > i < o > n
9 > 8 > 6 < 7 > 1 < 3 < 4 < 5 < 6 > 3 > 2 > 1 > 0 < 8 > 7
Comparison signs are printed for clarity. There exist other suitable chains for the word "`counterquestion`".

Problem D. Galaxy Center *(Division 2)*

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Ernest and Celestine arrived to the Galaxy Center in order to register their tiny house on the outskirts of the Galaxy. The Center is a vast system of halls and corridors. Ernest and Celestine appeared in the center hall and took a look at the neighborhood map on the wall.



The halls of the Center are separated into levels: level 0 consists of a single center hall, level 1 contains three halls, level 2 has nine halls, the third level contains as much as 27 halls, and so on. The Galaxy Center is really vast, so the number of levels can be considered infinite. The halls of each level except level 0 are connected by corridors to form a ring. Additionally, each hall is connected by a corridor to a hall of the next level. Conversely, every third hall is connected to a hall of the previous level. The corridors form a regular pattern as shown on the map above. Each corridor can be traversed in both directions. Amazingly, moving along any corridor in any direction takes the same fixed amount of time.

The travel routes in the Center are written as strings of lowercase English letters. Each letter denotes moving along a corridor to the neighboring hall. The letters correspond to the following directions on the map: "`s`" (spin) is counter-clockwise, "`a`" (antispin) is clockwise, "`c`" (centrifugal) is away from the center, and "`g`" (gravitational) is towards the center. An address of a hall is any route starting in the center hall and leading to this hall.

Ernest and Celestine have to first have a document signed in the hall at address s, and then seal it in the hall at address t. They have already completed the first part of the task, and are now located at address s. Help them find the shortest route to the address t. The length of a route is the number of letters in it.

Input

The first line contains the address s, and the second one contains the address t. Each of them contains from 1 to 35 letters. It is guaranteed that both addresses are valid routes from the center hall, and the halls at addresses s and t are distinct.

Output

Print the shortest route from the hall at address s to the hall at address t. If there are several possible answers, print any one of them.

Examples

standard input	standard output
ccc csss	gg
cccsca cccacs	aaaa

Problem E. IBM 1403 *(Division 2)*

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

The IBM 1403 line printer was introduced as part of the IBM 1401 computer in 1959 and had an especially long life in the IBM product line. (`https://en.wikipedia.org/wiki/IBM_1403`)

This printer works as follows. It receives an infinite paper tape which is exactly N characters wide (unfortunately, the printer supports monospaced printing only). The printing is done with a cyclic chain consisting of L characters. In the beginning, the chain is placed so that the characters numbered from 1 to N are located right above the first line of the paper.

The chain rotates with a constant speed of 1000 characters per second, so after one millisecond, the first line will be covered with characters from 2 to $N + 1$, and after L milliseconds, the chain will return to the starting position.

There are two actions used for printing. The first one is to print some characters. At any moment, the printer can send a signal to any subset of N positions on the paper, and after that, the selected subset of hammers strike the paper and make the corresponding characters from the chain appear on it. Of course, this action is allowed only at multiples of a millisecond, no one wants to print a character in the wrong place. The second action is to roll the paper forward by one line. A reverse roll is not possible, so the printer will do it only after the current line is ready (all necessary characters are printed). Striking the hammers can be considered instant, but rolling takes one millisecond. Remember that the chain still rotates while rolling.

Of course, the printer utilizes these actions in such a way that the total time of printing is minimized. Note that only one character can be printed in each position: for example, we can not print "`F`", then print an underscore in the same place and pretend that we printed an "`E`". However, the printer is allowed to leave some positions empty: a space is a *non-printable* character because it must not be printed.

Vasya, a Junior Assistant at VLLD (Very Long Logs Department), has just sent to the printer his debug output for the program which calculates working time for other programs. The output is huge, and it will take a lot of time to print. So Vasya decided to go outside and play snowballs with friends. But he wonders when he should come back to take the result. Help him determine when the printer will finish the task. It is guaranteed that the chain contains every character from Vasya's output at least once.

Please submit a program that, given a chain configuration and the text to print, determines the time it will take to print this text (the amount of time until the moment when the printer rolls the last line).

Input

The first line of input consists of L characters: the configuration of the chain. It contains the characters in the order they appear on the chain, starting with the character that is placed above position 1 of the paper in the beginning. Each of the following lines contains exactly N characters, they contain the text to be printed, line by line.

Constraints:

- $1 \le N \le L \le 10^5$,
- the chain may contain any characters with ASCII codes from 33 («`!`») to 126 («`~`»),
- the text may contain spaces (ASCII code 32) and any characters from the chain,
- the text is no more than two mebibytes long (2 097 152 bytes including the newline characters after each line), where each newline character is counted as two bytes for compatibility,
- the text may contain multiple consecutive spaces, heading spaces, and trailing spaces.

Output

Output one number: the time in milliseconds it will take the printer to print the given text.

Examples

standard input	standard output
qwertyuiopasdfghjklzxcvbnm1234567890-! sankt-peterburg ␣-␣samyj␣tramva jizirovannyj␣go rod␣v␣mire!␣!␣!	148
abc acc cab cab	6
++++++++++ ␣␣␣␣␣␣␣␣␣␣ ␣␣*␣␣␣␣␣*␣␣ ␣␣␣*␣␣␣*␣␣␣ ␣␣*******␣␣ ␣**␣***␣**␣ ********** *␣*******␣* *␣*␣␣␣␣␣*␣* ␣␣␣**␣**␣␣␣	14

Note

All space characters in the examples are shown as "␣" for better visibility.

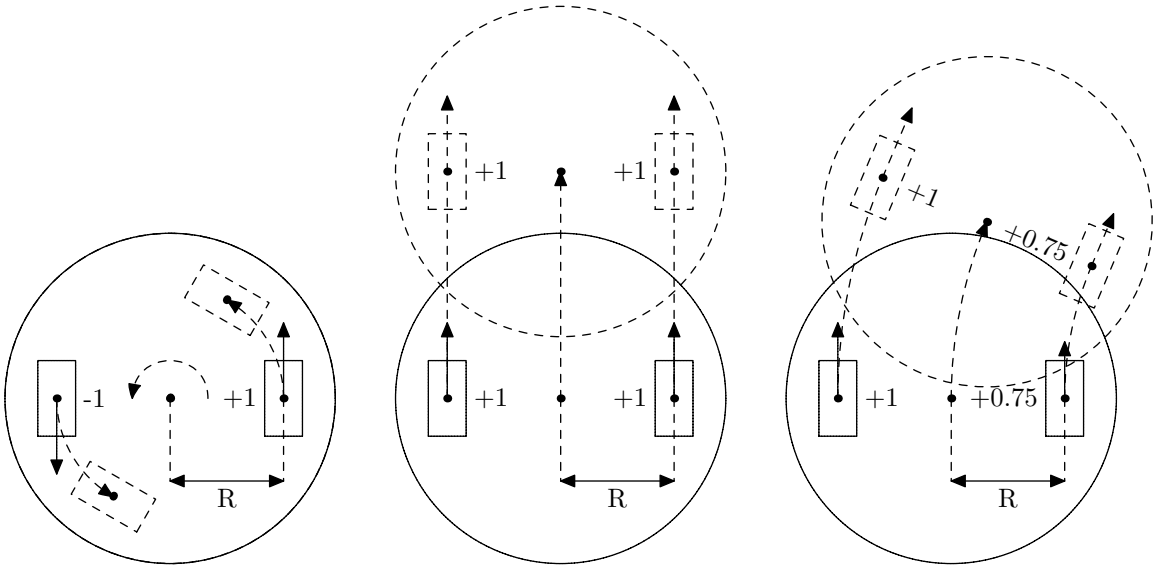
Problem F. Line Tracing *(Division 2)*

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

This is an interactive problem.

A typical robot vacuum cleaner is a *differential wheeled robot*. Such robots have two wheels of the same radius, which are fixed $2R$ units apart. Wheels may rotate independently of each other.

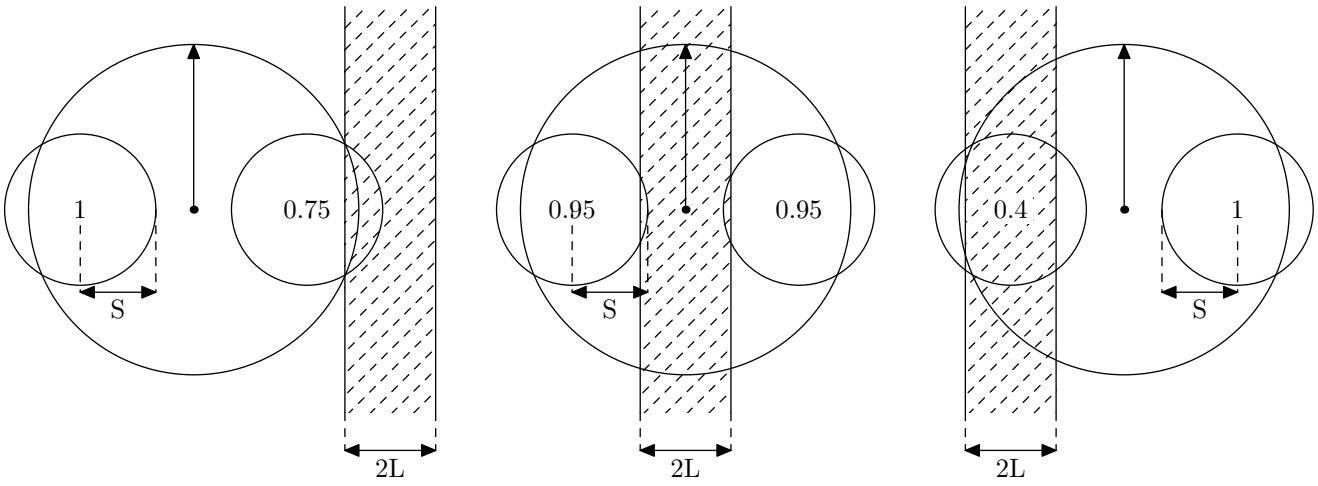
In this problem we assume that a motor can rotate a wheel with an arbitrary real speed between -1 and $+1$, inclusively. Depending on the difference between wheels' speed and rotation direction a robot would either go forward or turn:



A robot is located on an infinite white plane, on which a black polyline of width $2L$ is drawn. The robot starts in the beginning of that polyline and should go along it until the ending point.

The robot has two sensors; each is fixed exactly below a wheel. A sensor allows one to approximate the ratio of black color in a circle of radius S around the sensor. Each sensor yields a single real value between 0 (the circle is fully black) and 1 (the circle is fully white). Say, 0.5 means "approximately a half of the circle is black":

Sensors reading are also randomized on each step as follows: a mean of K uniform random numbers between -0.1 and 0.1 is added to each sensor's reading on each step (random numbers are chosen independently for each sensor on each step). Afterwards, if the reading exceeds 1, it's set to 1. Similarly if it's below -1, it's set to -1.



You have to program the robot so it will follow a polyline on the plane.

Originally robot's center is located right above the polyline's beginning (its first vertex) pointing to $\alpha + \beta$ (in radians), where α is the direction to the second vertex of the polyline, and β is a uniform random number between -0.2 and 0.2.

Step-by-step emulation follows. On each step your program should read robot's sensors' readings and print the desired wheel rotation speeds in order to continue. We assume that sensor's reading depend on its distance from the polyline d only, and the reading is always

$$1 - \frac{\max(0, \min(S, d + L) - \max(-S, d - L))}{2S}.$$

After receiving wheel rotation speeds c_l and c_r from your program, the jury will:

1. Add to each value a mean of K uniform random numbers between -0.1 and 0.1.
2. Replace c_l with $\min(\max(c_l, -1), 1)$, similarly for c_r.
3. Repeat T times: move robot by $\frac{(c_l + c_r) \cdot V}{2T}$ units in the direction of travel, and then turn it around by $\frac{(c_r - c_l) \cdot V}{2RT}$ radians.
4. Go to the next step of emulation.

Should the robot move away from the polyline farther than $R + S + L$, your solution gets "`Wrong Answer`". If your solution does not *pass* the polyline after M steps, it gets "`Wrong Answer`". A polyline of n vertices is considered *passed* if there are step numbers $t_1 < t_2 < \cdots < t_n$ such that on step t_i the robot was closer than $R + S + L$ to the vertex p_i of the polyline.

All tests for this problem can be downloaded here: `http://acm.math.spbu.ru/171217/linetracing.zip`. Each file starts with a single integer, number of polyline's vertices, followed by coordinates of vertices.

Constraints

$R = 1.5$, $S = 1.0$, $L = 0.6$, $K = 10$, $V = 0.5$, $T = 100$, $M = 20\,000$, the total length of the polyline does not exceed 5000.

Interaction Protocol

In the beginning of each step, your program receives a separate line with two reals between 0 and 1 each: readings of left and right sensors, correspondingly. Afterwards, you should print two reals between -1 and $+1$ each: the desired wheel rotation speeds after the current emulation step.

To prevent output buffering, flush the output buffer after each line: this can be done by using, for example, `fflush (stdout)` in C or C++, `System.out.flush ()` in Java, `flush (output)` in Pascal,

or `sys.stdout.flush()` in Python. Also, do not forget to terminate each line of output with a newline character.

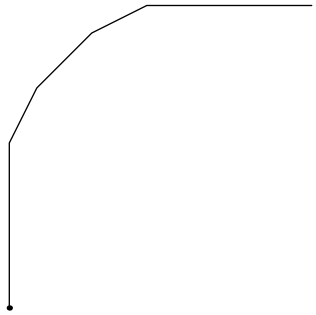
If your robot passes the polyline, your program receives "`-1 -1`" as the input instead of sensors' readings. In that case, the program should immediately terminate gracefully.

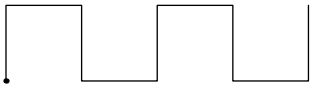
Example

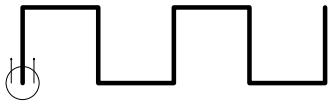
standard input	standard output
0.96364308 0.94457910 0.99295765 0.95889118 0.99212102 0.90060966 ... -1 -1	 1 0.8 0.9 1 1 0.5 ... 1 1

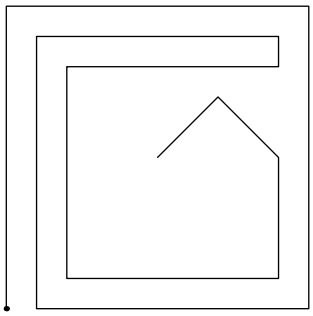
Notes

Tests 1–3 are straight lines of lengths 10, 100 and 4000, correspondingly. The remaining tests are pictured below.

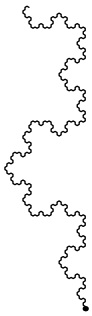

#4;W=110;H=110

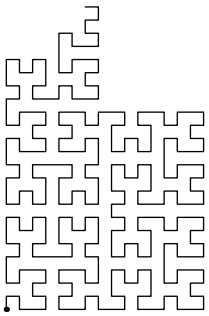

#5;W=400;H=100


#6;W=40;H=10


#7;W=100;H=100


#8;W=999;H=50.00024


#9;W=346.41022;H=1305


#10;W=150;H=230


#11;W=48.11299;H=7.98282


#12;W=220.98079;H=855.8216


#13;W=1766.66559;H=454.65488


#14;W=757.19681;H=2004.82149


#15;W=678.20271;H=1167.25923

Problem G. The Queen and the Knight *(Division 2)*

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	256 mebibytes

This is an interactive problem.

Once upon a time there lived a Queen and a Knight. They were happy until the Knight has stolen some money from the Kingdom's Treasury. The Queen now wants to capture the Knight and put him in jail.

Imagine that it all happens on the chessboard $N \times N$. The Queen and the Knight are the corresponding pieces and move according to the standard chess rules: the Queen moves an arbitrary non-zero number of squares vertically, horizontally, or diagonally; the Knight makes an L-shaped move, changing one of the coordinates by 1 and another coordinate by 2.

Your program will be playing as the Queen, and the jury program will be playing as the Knight.

Your task is to capture the Knight in at most $4N$ moves. The Queen moves first.

Interaction Protocol

To pass a single test, your solution must win K scenarios ($1 \le K \cdot N \le 10\,000$). All scenarios in a single test take place on the same board of size $N \times N$.

The input starts with a single line containing a single integer N ($4 \le N \le 100$). The scenarios follow.

Each scenario is started by specifying the initial positions of the Queen and the Knight. They are given on a single line as four integers separated by spaces: Qx, Qy, Kx and Ky, where $1 \le Qx, Qy, Kx, Ky \le N$. It is guaranteed that the Queen and the Knight start on different squares.

After that, you need to make your moves. Each move is made by printing a single line containing two space-separated integers: Qnx and Qny, the new position you selected for the Queen ($1 \le Qnx, Qny \le N$). It is allowed to make only valid Queen moves, in particular, it is not allowed to stay on the same square. The jury program responds with a single line containing two integers: Knx and Kny, the new position for the Knight ($1 \le Knx, Kny \le N$). It is guaranteed that the Knight makes only valid moves.

If one of the pieces moves to the position of the other, the latter piece is captured, and the former one wins the game. If you have captured the Knight (you won), or the Knight can capture you on his move (you lost), or you already made $4N$ moves and the game did not finish (you lost), or you made an invalid move (you lost), the interactor responds with two zeroes instead of a Knight's move. In this case, your should read the next scenario.

For each test, the scenarios are fixed in advance and do not change during the contest. Obviously, each test is only passed if you won all scenarios in that test.

After the last scenario in the test, or after the first scenario where you lost (whichever comes first), there will be a line containing four zeroes instead of the positions. After reading this line, your program should immediately terminate gracefully.

To prevent output buffering, flush the output buffer after each move: this can be done by using, for example, `fflush (stdout)` in C or C++, `System.out.flush ()` in Java, `flush (output)` in Pascal, or `sys.stdout.flush ()` in Python. Also, do not forget to terminate each line of output with a newline character.

Example

standard input	standard output
4 1 1 2 3 3 1 0 0 1 1 1 4 0 0 0 0 0 0	 2 2 3 1 3 3

Note

Note that in the sample above the program lost the second scenario. Your program could consider taking the Knight instead of moving the Queen on a square where the Knight can capture it.

Problem H. Product of Roots *(Division 2)*

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	256 mebibytes

Bob loves to play with polynomials. Yesterday he came up with two pairs of non-negative integers: a_1, a_2 and b_1, b_2.

After that, he constructed polynomials

$$f(x) = (1 + a_1 x)(1 + a_2 x),$$

$$g(x) = (1 + b_1 x)(1 + b_2 x),$$

$$h(x) = (1 + a_1 b_1 x)(1 + a_2 b_1 x)(1 + a_1 b_2 x)(1 + a_2 b_2 x).$$

Bob loves not only polynomials, but also Fourier transform. That is why he calculated coefficients of his polynomials modulo 998 244 353.

Unfortunately, today Bob has lost his integers a_1, a_2, b_1, b_2, along with the polynomial $h(x)$. He has only polynomials $f(x)$ and $g(x)$ left. Bob has got upset and now asks Alice to help him restore polynomial $h(x)$ knowing only polynomials $f(x)$ and $g(x)$. Alice has no time to deal with the problem, so she asks you to solve it instead.

Input

The first line contains integer coefficients f_0, f_1, f_2, where $f(x) = f_0 + f_1 x + f_2 x^2$.

The second line contains integer coefficients g_0, g_1, g_2, where $g(x) = g_0 + g_1 x + g_2 x^2$.

All coefficients are given modulo 998 244 353.

Output

Output exactly five integer coefficients h_0, h_1, h_2, h_3, h_4, where $h(x) = h_0 + h_1 x + h_2 x^2 + h_3 x^3 + h_4 x^4$.

The coefficients must be taken modulo 998 244 353.

Example

standard input	standard output
1 2 1 1 2 1	1 4 6 4 1

Problem I. Safe Landing *(Division 2)*

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

A giant humanoid battle robot Georgiy stands on a flat plateau. The plateau is divided into squares aligned with cardinal directions: the grid lines go from south to north and from west to east. Georgiy is fully contained in a square. We will consider the plateau as infinite in all directions.

Marcella the Diplomat hovers above the plateau in a helicopter. She wants to land and enter negotiations with Georgiy. But landing may be unsafe: Georgiy can trample the helicopter as soon as it lands, before negotiations can even start!

Georgiy can move along the plateau: in one move, he goes from a square to another one which has a common side. Fortunately, after a recent battle, Georgiy's navigation system is damaged: he can move only in three cardinal directions and can not move in the fourth direction. Unfortunately, it is not known which side is the one in which he can not move.

Marcella has observed Georgiy for some time and recorded all his moves. Surely, the records can not contain all four cardinal directions. At the moment, Marcella hovers right above Georgiy. Help her choose a safe square to land: such square that Georgiy is not there right now and certainly can not come there, or determine that the records do not allow to choose such a square with certainty.

Input

The first line contains some characters without spaces: the records of Georgiy's moves. Each character is a capital English letter denoting a cardinal direction in which the robot moved: "`N`" (north), "`W`" (west), "`S`" (south), or "`E`" (east). It is guaranteed that the line contains at most three different directions. The line ends with a newline and contains between 1 and 100 characters.

Output

If there exists a square such that Georgiy is not there right now and certainly can not come there, output a route for Marcella's helicopter which starts at where Georgiy is now and ends in such safe square. The route must be printed in the same format as the input route, and it must contain between 1 and 100 characters. If there are multiple such routes, print any one of them. In case no square of the plateau is certainly safe, print one capital letter "`X`" (ex) instead of a route. The output may end with a newline.

Examples

standard input	standard output
NNWNEE	SES
S	X

Explanations

In the first example, the records show Georgiy moving north, west and east. Consequently, he certainly can not move south. Any cell to the south of Georgiy's current position is safe. For instance, Marcella can fly from Georgiy's position to the south, to the east and again to the south, as shown in the example.

In the second example, it is only known that Georgiy can move south. Unfortunately, this leaves three options for the direction in which he can not move, and consequently, no cell is certainly safe. For instance, if Marcella picks the route "`N`", it may happen that Georgiy can not move east but can move north. If the route "`NEE`" is chosen, it may happen that Georgiy can not move west but can move north and east, and so on.

Problem J. Perfect Square *(Division 2)*

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

An integer n is a perfect square if there exists an integer x such that $x \cdot x = n$.

The integer n is given. Determine whether it is a perfect square or not. Keep in mind that n can be very large!

Input

The first line contains a positive integer n in decimal notation. It is guaranteed that this number contains from 1 to 1 000 000 digits, and the first digit is positive.

Output

Print "`Yes`" if n is a perfect square, or "`No`" otherwise. The case of letters does not matter: for example, you can print "`yES`" or "`NO`".

Examples

standard input	standard output
25	Yes
27	No