Problem A. Appearance Analysis

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

You have taken a photo of the Faculty of Electrical Engineering and Computing "C" building here in Zagreb with its equal sized windows neatly arranged in rows and columns. Some of the windows are painted in curious designs and you are trying to analyze the photo and determine the number of distinct designs.

We represent the photo as a rectangular grid of characters with r rows and c columns. Every window is a rectangular area and all windows have the same dimensions. Each cell in a window is either clear (denoted by the "." character) or painted (denoted by the "+" character). Two windows are considered to be of the *same design* if one can be rotated a multiple of 90 degrees and placed on top of the other so that they perfectly match. When comparing designs, we are not allowed to flip a window inside out.

Windows are regularly aligned in rows and columns with exactly one row of brick cells (denoted by the "#" character) framing each window. More precisely, there is a single row of "#" characters between two consecutive window rows as well as before the first window row and after the last window row. Similarly, there is a single column of "#" characters between two consecutive window columns as well as before the first window column and after the last window column. The exact number of window rows and window columns is arbitrary. The window dimensions are also arbitrary. However, a window consists of at least one cell and, again, all windows in the photo have the same dimensions.

Find the number of different window designs in the photo.

Input

The first line contains two integers r and c ($3 \le r, c \le 111$) — number of rows and the number of columns of the photo. Each of the following r lines contains a string consisting of c characters — one row of the photo.

Output

Output a single integer — the number of different window designs in the photo.

Examples

standard input	standard output
11 16 ################ #....#++++#+...# #....#++.+#+...# #....#.++.#++.+# #....#....#++++# ################ #....#.+..#++++# #..++#.+..#++.+# #+...#....#.++.# #+...#..++#....# ################	4
9 21 ###################### #...+#++++#+...#..+.# #..+.#.++.#.+..#..+.# #.+..#....#..+.#..+.# ###################### #.+..#....#..+.#.+..# #..+.#.++.#.+..#.+..# #...+#++++#+...#.+..# ######################	4

Problem C. Convex Contour

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

A number of geometric shapes are neatly arranged in a rectangular grid. The shapes occupy consecutive cells in a single row with each cell containing exactly one shape. Each shape is either:

- a square perfectly aligned with the grid square,
- a circle inscribed in the grid square,
- or an equilateral triangle with a side corresponding to the bottom side of the grid square.

At the pictures below you can see the shapes from the first example input and their convex contour.

The shapes from the first example input and their convex contour.

Informally, the *convex contour* of an arrangement is the shortest line that encloses all the shapes. Formally, we can define it as the circumference of the convex hull of the union of all shapes.

Given an arrangement of shapes, find the length of its contour.

Input

The first line contains an integer n ($1 \le n \le 20$) — the number of shapes. The following line contains a string consisting of n characters that describes the shapes in the arrangement left to right. Each character is an uppercase letter "`S`", "`C`" or "`T`" denoting a square, circle or a triangle respectively.

Output

Output a single floating point number — the length of the contour. The solution will be accepted if the absolute or the relative difference from the judges solution is less than 10^{-6}.

Example

standard input	standard output
4 TSTC	9.088434417
3 SCT	7.50914177324

Problem E. Easy Equation

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Given an integer k greater than 1, it is possible to prove that there are infinitely many triples of positive integers (a, b, c) that satisfy the following equation:

$$a^2 + b^2 + c^2 = k(ab + bc + ca) + 1.$$

Given positive integers n and k, find n arbitrary triples (a_1, b_1, c_1), (a_2, b_2, c_2), $\ldots$, (a_n, b_n, c_n) that all satisfy the equation. Furthermore, all $3n$ integers $a_1, \ldots, a_n, b_1, \ldots, b_n, c_1, \ldots, c_n$ should be different positive integers with at most 100 decimal digits each.

Input

The first line contains two integers k and n ($2 \le k \le 1\,000, 1 \le n \le 1\,000$) — the constant k in the equation and the target number of triples.

Output

Output n lines. The i-th line should contain three space separated integers a_i, b_i and c_i with at most 100 digits each — the i-th of the solutions you found.

Examples

standard input	standard output
2 8	1 2 6 3 10 24 12 35 88 15 28 84 4 5 18 14 33 90 40 104 273 21 60 152
3 3	1 3 12 8 21 87 44 165 615

Problem F. Free Figurines

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

A *matryoshka doll* is a set of wooden figurines of increasing sizes that can be nested one inside the other. They are nested by placing a figurine inside a larger figurine which is, in turn, placed inside a yet larger figurine, etc. We may never place more than one figurine *directly* inside another figurine, no matter the sizes of the figurines involved.

We are currently playing with a set of n figurines denoted by integers 1 through n in the order of increasing size. If a figurine a is placed directly inside a figurine b we say that b is a *parent* of a, and if a figurine has no parent we call it *free*. A *configuration* of the whole set can be described by specifying the current parent of each figurine.

We are allowed to perform the following steps while playing:

- Place a free figurine inside a larger free figurine that is currently empty.
- Open a non-empty free figurine and take out the figurine placed directly inside.

Find the minimal number of steps to obtain the given target configuration from the given initial configuration.

Input

The first line contains an integer n ($1 \le n \le 100\,000$) — the number of figurines.

The following line contains a sequence of n integers $p_1, p_2, \ldots, p_n$ ($0 \le p_k \le n$) describing the initial configuration. The k-th integer p_k is 0 if the figurine k is free, or the parent of figurine k otherwise.

The following line contains a sequence of n integers $q_1, q_2, \ldots, q_n$ ($0 \le q_k \le n$) describing the target configuration in the same format.

You may assume that both configurations are valid: a figurine is always smaller than its parent and no two figurines have the same parent.

Output

Output a single integer — the minimal number of steps.

Examples

standard input	standard output
7 3 5 4 0 7 0 0 3 5 0 6 7 0 0	2
6 2 5 4 0 0 0 2 6 4 5 0 0	3

Problem G. Hangar Hurdles

Input file:	*standard input*
Output file:	*standard output*
Time limit:	8 seconds
Memory limit:	512 mebibytes

You are evaluating constructions plans for a giant new hangar that will house an airplane assembly line. The hangar floor can be represented as a rectangular grid consisting of n rows and n columns where every cell is either empty or blocked. The rows are numbered with integers 1 through n top to bottom, while the columns are numbered with integers 1 through n left to right.

It is important that large crates containing airplane parts can be freely moved between various locations inside the hangar. We can model a crate as a square aligned with grid cells and centered in one of the cells. Therefore, for an odd integer k, a *crate of size* k consists of cells in k consecutive rows and k consecutive columns. The crate can be moved up, down, left or right as long as it fits completely inside the grid and never contains a blocked cell.

You are given q pairs of cells A_k and B_k. For each pair, find the size of the largest crate that can be centered in A_k and then moved across the hangar floor until it's centered in B_k.

Input

The first line contains an integer n ($2 \le n \le 1\,000$) — the size of the hangar floor. Each of the following n lines contains a string of exactly n characters describing one row of the floor. The character "#" denotes a blocked cell while the character "." denotes an empty cell.

The following line contains an integer q ($1 \le q \le 300\,000$) — the number of queries. The k-th of the following q lines contains four integers $r_{A_k}, c_{A_k}, r_{B_k}, c_{B_k}$ ($1 \le r_{A_k}, c_{A_k}, r_{B_k}, c_{B_k} \le n$) — the row number and column number of cells A_k and B_k respectively. Cell A_k will be different than the cell B_k. Also, both cells will always be empty.

Output

Output q lines. The k-th line should contain a single integer s_k — the size of the largest crate that can be moved from A_k to B_k. If no crate can be moved from A_k to B_k then s_k should be 0.

Example

standard input	standard output
7 .....#. ...#.#. ....#.. ....### ....#.. #...... .... 5 2 5 5 2 2 5 3 6 2 2 6 3 2 2 6 6 1 1 7 7	1 0 3 1 1

Problem K. Key Knocking

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Goran is recovering from his knee surgery and is experimenting with a smart card used for storing cryptographic keys. In this problem, a *key* is a bitstring of size $3n$ where n is a positive integer. Particular bits of the key are indexed with integers 1 through $3n$ left to right. *Weight* of a key is the number of pairs of different neighbouring bits increased by one. For example, the weight of the key "000" is 1, while the weight of the key "011010100" is 7.

He has discovered that he can tamper with the key by sending small jolts of electricity through the circuits of the smart card. In particular, he can reliably perform the following operation: pick two arbitrary neighbouring bits of the key and flip both of them. For example, one operation can change the key "000" to "110".

Given a key of size $3n$ find any sequence of at most n operations that transforms the given key to a key of weight at least $2n$. You may assume that a solution always exists.

Input

The first line contains a string consisting of digits "0" and "1" — the initial key. The length of the key is $3n$ where n is a positive integer such that $1 \le n \le 100\,000$.

Output

The first line should contain an integer m where $0 \le m \le n$ — the number of operations in your solution. The following line should contain m integers $a_1, a_2, \ldots, a_m$ describing your solution. The number a_k is the index of the left one of two bits being flipped in the k-th step.

If the initial key already has weight at least $2n$ you may output only the integer 0.

Examples

standard input	standard output
000000000	3 2 5 6
111001000111	2 3 9
010101	0

Problem M. Mighty Monster

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

In the game «Heroes of the Fight and Rage» you are fighting dangerous monster. The monster has the following moves:

- Rake, denoted by the letter '`R`';
- Bite, denoted by the letter '`B`';
- Laser breath, denoted by the letter '`L`'.

In order to defend, you must perform a counter move per move that the monster makes:

- Slice, denoted by the letter '`S`', counters the monster's rake;
- Kick, denoted by the letter '`K`', counters the monster's bite;
- Shield, denoted by the letter '`H`', counters the monster's laser breath;

However, there is one catch. When the monster performs a subsequent combination of the three moves Rake, Bite and Laser breath, in any order, it becomes a very powerful attack for which you must perform a single counter move called Combo breaker, denoted by the letter '`C`'. A single Combo breaker absorbs the entire combination of three moves. Any following moves from the monster will have to be countered separately or as part of a new combination. A move of the monster can never be part of more than one combination.

Given a string, representing the monster moves, print the sequence of your actions to counter all monster's attacks.

Input

A single line containing a string of at least 1 and at most 10^6 characters, consisting of the letters '`R`', '`B`' and '`L`'.

Output

Output a single string consisting of the letters denoting the moves that are to be made in succession in order to defend from the monster's attacks.

Example

standard input	standard output
RRBBBLLR	SSKKKHHS
RBLLLBRR	CHCS
RBLBR	CKS

Problem N. Needle Navigator

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

The API of the compass simulator is next: when the compass needle is currently in some direction (between 0 and 359 degrees, with north being 0, east being 90), it is being animated by giving the degrees to spin it.

For example, if the needle is pointing north, and you give the compass an input of 90, it will spin clockwise (positive numbers mean clockwise direction) to stop at east, whereas an input of -45 would spin it counterclockwise to stop at north west.

Your task is to find the *shortest path* (angle) from the current needle direction to the correct direction.

Input

The first line of input contains an integer n_1 ($0 \le n_1 \le 359$), the current direction of the needle. The second line of input contains an integer n_2 ($0 \le n_2 \le 359$), the correct direction of the needle.

Output

Output the change in direction that would make the needle spin the shortest distance from n_1 to n_2. A positive change indicates spinning the needle clockwise, and a negative change indicates spinning the needle counter-clockwise. If the two input numbers are diametrically opposed, the needle should travel clockwise.

Example

standard input	standard output
315 45	90
180 270	90
45 270	-135

Problem O. Make More Money!

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Foretelling the future is hard, but imagine if you could just go back in time and use your knowledge of stock price history in order to maximize your profits!

Lets you can go back in time a few days and bring a measly 100 dollars with you. How much money could you make by just buying and selling stock of one company at the right times?

Note that you can not buy fractional shares, you must buy whole shares. The total number of shares is 100 000, so you can not own more than 100 000 shares at any time.

For simplicity, there are no fees for buying and selling stocks, stock prices change only once per day, and your trading does not influence the valuation of the stock.

Input

The first line of input contains an integer d ($1 \le d \le 365$), the number of days that you can go back in time. Then follow d lines, the i'th of which contains an integer p_i ($1 \le p_i \le 500$) giving the price at which you can buy or sell stock on day i. Days are ordered from oldest to newest.

Output

Output the maximum possible amount of money you can have on the last day. Note that the answer may exceed 2^{32}.

Example

standard input	standard output
6 100 200 100 150 125 300	650

Problem P. Several Sticks

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Given n sticks of integer length. Find out if you can build the non-generated triangle, using three of those sticks.

Input

The first line contains a single integer N, with $3 \le N \le 2 \cdot 10^4$ — the number of sticks collected. Then follows one line with N positive integers, each less than 2^{60}, the lengths of the sticks you have.

Output

Output a single line containing a single word: "`possible`" if you can make a triangle with three sticks of the provided lengths, and "`impossible`" if you can not.

Example

standard input	standard output
3 1 1 1	possible
5 3 1 10 5 15	impossible

Problem Q. Falling Stones

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Consider a 2D grid, which contains stones, obstacles, and open spaces. Gravity will pull the stones straight down, until they hit an obstacle, or the bottom of the grid, or another stone which has already come to rest. Obstacles don't move. Given such a grid, determine where the stones eventually settle.

Input

The first line of input contains two integers, r and c ($1 \le r, c \le 100$), which are the number of rows and the number of columns of the grid. On each of the next r lines will be c characters: '`o`' for a stone, '`x`' for an obstacle, and '`.` for an open space.

Output

Output the grid, after the stones have fallen.

Examples

standard input	standard output
4 4 oooo x... ..x. x.xx	o... x.o. ..xo xoxx
4 3 o.o o.o o.. ...	... o.. o.o o.o

Problem R. Regular Strings

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Define a *k-regular* string as follows:

A string s is k-regular if the length of the string $|s|$ is a multiple of k, and if you chop the string up into $|s|/k$ substrings of length k, then each of those substrings (except the first) is the same as the previous substring, but with its last character moved to the front.

For example, the following string is 3-regular: "`xyzzxyyzxxyz`" The above string can break up into substrings "`xyz`", "`zxy`", "`yzx`", and "`xyz`", and each substring (except the first) is a right-rotation of the previous substring.

Given a string, determine the smallest k for which the string is k-regular.

Input

The single line of input contains a string s ($1 \le |s| \le 100$) consisting only of lowercase English letters.

Output

Output the integer k, which is the smallest k for which the input string is k-regilar

Examples

standard input	standard output
qqqqqq	1
cddccddccddc	2
amppz	5