Problem A. Z-function to prefix-function

Time limit: 1 second
Memory limit: 256 megabytes

You are given z-function of some (unknown for you) string s.

Write prefix-function of the string s.

Input

The first line contains one integer: n ($1 \le n \le 2 \cdot 10^5$) — the length of string s.

The second line contains z-function — n space-separated integers.

Output

Write n space-separated numbers — prefix-function of the string s.

Examples

standard input	standard output
8 0 0 1 0 3 0 1 1	0 0 1 0 1 2 3 1

Note

In the example s is equal to "ABACABAA".

Problem B. Prefix-function to z-function

Time limit: 1 second
Memory limit: 256 megabytes

You are given prefix-function of some (unknown for you) string s.

Write z-function of the string s.

Input

The first line contains one integer: n ($1 \le n \le 2 \cdot 10^5$) — the length of string s.

The second line contains prefix-function — n space-separated integers.

Output

Write n space-separated numbers — z-function of the string s.

Examples

standard input	standard output
8 0 0 1 0 1 2 3 1	0 0 1 0 3 0 1 1

Note

In the example s is equal to "ABACABAA".

Problem C. Prefix-palindromes

Time limit: 0.5 seconds
Memory limit: 256 megabytes

You problem is to find the length of the longest prefix which is a palindrome at the same time.

Input

The only line contains s, consisting of lowercase Latin letters. The length doesn't exceed 10^5.

Output

Print the only number — the length of the longest such prefix that is a palindrome. If the whole string is palindrome you should print its length.

Example

standard input	standard output
aaab	3
abab	3

Problem D. Words

Time limit: 0.5 second
Memory limit: 256 megabytes

For given string $s = s_1 s_2 \dots s_n$ the allowed operation consists of the following steps:

- choose integer value k between 0 and n, inclusive;
- remove first k characters, but append them in the reverse order: s becomes $s_{k+1} \dots s_n s_k s_{k-1} \dots s_1$.

For two given strings s and t you are to check if exists such operation that the result equals to t. You can apply an operation only once.

Input

The input contains two lines: s and t. They have lengths between 1 and 50000, inclusive. They can have different lengths.

Both strings contain only uppercase and lowercase English letters and digits.

Output

Print "Yes" if the required operation exists, and "No" if not. In case of positive answer print possible value of k. If there many valid values of k, use the minimal.

Examples

standard input	standard output
wpwdwpw wdwpwpw	Yes 2

Problem E. Inaccurate Search

Time limit: 2 seconds
Memory limit: 256 megabytes

Given text $t = t_1 t_2 \dots t_{|t|}$ and pattern $p = p_1 p_2 \dots p_{|p|}$. Also given the parameter k.

For position d if $t[d \dots d + |p| - 1] = p$ then there is an exact matching from the position d.

For position d if in substring $t[d \dots d + |p| - 1]$ there is a window (segment) of f consecutive characters which can be changed to make the substring be equal to p and $f \le k$ then it is *inaccurate matching*. Obviously, an exact matching is special case of an inaccurate matching.

For example, if $k = 3$, t="abacabadabacaba" and p="baxayad", there is one inaccurate matchings from the position 2 (indices are 1-based).

Your task is to find all inaccurate matchings of p for given text t.

Input

The first line contains the text t. The second line contains the pattern p. Both of them contain only Latin letters. Their lengths are between 1 and 10^6, inclusive. The third line contains k ($0 \le k \le 10^6$).

Output

Print the number of inaccurate matchings and the corresponding positions in increasing order.

Examples

standard input	standard output
abacaaa aaa 1	4 1 3 4 5
ababa aaaaa 3	1 1

Problem F. Fibonacci Strings

Time limit: 2 seconds
Memory limit: 1024 megabytes

Memory limit is 1024 megabytes.

The sequence $f_0, f_1, \ldots$ is Fibonacci Strings if:

- $f_0 =$ “”
- $f_1 =$ “a”
- $f_2 =$ “b”
- $f_3 =$ “ab”
- $f_4 =$ “bab”
- ...
- $f_i = f_{i-2} + f_{i-1}$

Given string s containing only letters ‘a’ and ‘b’ и and non-negative integer k.

Find the number of occurrences string s in the string f_k modulo 1000000009 ($10^9 + 9$).

Input

The first line contains non-empty s, the length of s is not greater than 10^3. It contains only ‘a’ and ‘b’.

The second line contains integer k ($0 \le k \le 10^4$).

Output

Print the number of occurrences modulo 1000000009 ($10^9 + 9$).

Examples

standard input	standard output
b 4	2
ab 5	2

Problem G. Placing Pieces

Time limit: 0.5 seconds
Memory limit: 256 megabytes

You have received a new puzzle as a gift. It consists of a board and several pieces of various lengths. Each piece has the same width as the board. The goal of the game is to place the least number of pieces on the board such that no other unused piece can be added legally. Pieces must not hang over the edge of the board or be twisted at an angle. Each piece must be oriented so that its width is parallel to the width of the board. Pieces must not overlap, but their edges may touch. Keep in mind that distances between pieces or the distances between pieces and the edges of the board are not necessarily integer numbers. See the examples for further clarification.

You will be given an integer l — the length of the board, and an array of integers containing the lengths of the pieces. Print the number of pieces placed on the board that allows you to solve the puzzle.

Input

The first line contains only one integer l ($1 \le l \le 1000$) — the length of the board.

The second line contains only one integer n ($1 \le n \le 30$) — the number of pices.

The third line contains n space-separated integers a_i ($1 \le a_i \le 100$) — the length of pices.

Output

Print only one integer — the number of pieces placed on the board that allows you to solve the puzzle.

Examples

standard input	standard output
9 2 1 8	1
36 5 1 1 5 5 5	4
37 5 1 1 5 5 5	5

Note

In the first example you can place both pieces on the board and leave no space on it. However, there is a better solution as depicted below. Place only the second piece on the board, and leave a little space between it and the left and right edges of the board, so that the first piece can't fit:

In the second example If we place all three pieces with length 5 on the board, we will have no choice but to also place the smallest

two pieces. However, if you place only two of them and the two smallest pieces, you can leave spaces on the board that are smaller than the length of the remaining piece.

Problem H. Kids Game

Time limit: 0.5 seconds
Memory limit: 256 megabytes

Little Johnny and his friends play a lot of games in which each player gets a different role. The roles are assigned using a method reminiscent of "eenie meenie miny moe"rhymes. n kids stand in a circle and are numbered from 1 to n going in a clockwise direction. They choose a number m, and starting with kid 1, they go around the circle in a clockwise direction, counting off from 1 to m. The kid who gets number m is eliminated from the circle, and the counting starts again at 1 with the next kid. The i-th eliminated kid gets the ith role in the game. Johnny wants to know what role he will get if he is kid number k in the circle.

For example, consider the case where $n = 5$, $m = 2$, and $k = 3$. The kids are arranged clockwise as follows: $1, 2, 3, 4, 5$. Starting with kid 1, they start counting from 1 to 2. Kid 2 gets number 2, so he is eliminated from the circle, which now looks like: $1, 3, 4, 5$. They start counting again with kid 3. Kid 4 gets number 2 this time, so he is the next to get eliminated. Then, kid 1 is eliminated, followed by kid 5, and finally, kid 3. Johnny is kid 3, so he is the 5-th kid to get eliminated, and he is assigned the 5-th role.

Given integers n, m and k, print the role assigned to Johnny. Roles are 1-indexed, so the 1st eliminated kid gets role 1, the 2nd eliminated kid gets role 2, and so on.

Input

The first line contains three integer n, m and k $(1 \le n, m, k \le 5 \times 10^5, k \le n)$.

Output

Print only one integer — the role assigned to Johnny.

Examples

standard input	standard output
10 3 6	2
10 3 4	10

Problem I. Binary Matrix

Time limit: 0.5 seconds
Memory limit: 256 megabytes

You are given a binary rectangular matrix with exactly five columns. A matrix is called good if it contains exactly a_i ones in the i-th column. To make the matrix good, you are allowed to perform at most m moves. With each move, you select a row of the matrix and circularly shift it to the right by 1 position (so, for example, row $0, 0, 1, 0, 0$ becomes $0, 0, 0, 1, 0$).

Print the lexicographically greatest good matrix you can achieve. If you can not achieve any good matrix, leave the output empty. To compare two matrices lexicographically, concatenate all rows starting from the top for each of the matrices and compare the resulting strings. String s is lexicographically greater than string b if it contains the bigger character at the first position where they differ.

Input

The first line contains only one integer n $(1 \le n \le 40)$ — the number of rows in matrix.

The following n lines contain one row on each. Each row contains exactly 5 characters. Each character is either '0' or '1'.

The following line contain exactly 5 space-separated integers a_i $(0 \le a_i \le 40)$ — the number of ones in i-th column of good matrix.

The last line contains only one integer k $(0 \le k \le 160)$ — the maximum number of moves.

Output

Print n lines with exactly 5 characters on each — the lexicographically greatest good matrix you can achieve or leave the output empty if you can not achieve any good matrix.

Examples

standard input	standard output
2 01000 10000 1 1 0 0 0 5	10000 01000
2 01000 10000 1 1 0 0 0 4	01000 10000
5 00100 10000 00010 00001 01000 1 1 1 1 1 7	10000 01000 00010 00001 00100

Note

The first example you need to shift the first row 4 times and the second row once.

The second example you don't change anything.

Problem J. Great Berland Wall

Time limit: 2 seconds
Memory limit: 256 megabytes

Berland is in civil war. This time peaceful solution is hardly possible. Collapse to the East and West Berland is inevitable! Commander-in-chief of grey, general Kruglyakovski, has made a decision to hasten the process and to surround his headquarters

with a wall (which later be called Great Berland Wall). The wall should be built in such a way that the enemy's headquarters would stay at the other side of the wall.

Looking to the Berland map general Kruglyakovski noticed that the country consists of a number of provinces. Each province has the form of a simple (but possibly not convex) polygon. The country of Berland has no enclaves and does not contain any enclaves of other countries inside, i.e. it is possible to get from one point of Berland to another without leaving the country. The "passability" is known for each segment of the border of each province. "Passability" is the maximum speed the soldier can move with along the corresponding segment.

General Kruglyakovski decided that the wall will pass only along the borders of the provinces and will have the form of a simple polygon. He sent the group of soldiers to build the wall during the night. And you got a task to find such a form and position of the wall that the time of building would be minimal. The time of building of the section of the wall is equal to the "passability" of the corresponding border segment. The time of building of the wall is equal to the sum of building of its segments.

Input

The first line of the input file contains the number of segments of province borders — integer number N ($5 \le N \le 300$). The segments themselves follow further. Each segment is defined by five numbers x_1, y_1, x_2, y_2, v, where (x_1, y_1) and (x_2, y_2) are the coordinates of the segment end points and the integer number v ($1 \le v \le 1000$) is the "passability". Any pair of segments has not more than one common point and this point can only be their common end-point. Input data finishes with one more quadruple of numbers X_1, Y_1, X_2, Y_2, where (X_1, Y_1), (X_2, Y_2) are the coordinates of headquarters of the general Kruglyakovski and his opponent correspondingly. Both headquarters are situated strictly inside one of the provinces. The headquarters are situated in different provinces. All coordinates in the input are integer numbers less than 10^4 by the absolute value.

Output

Write to the first line of the output the total time of building of the wall. Write to the second line the number of border segments the wall will occupy. To the next line write the sequence of the numbers of the segments. The segments are numbered in the order they appear in the input. If there are several solutions choose any of them.

Example

standard input	standard output
13 0 6 3 6 9 0 0 4 2 8 4 4 6 6 7 2 4 3 6 1 3 6 6 6 1 6 4 6 6 1 4 2 6 4 1 0 0 0 6 6 2 2 2 4 1 2 2 4 2 1 0 6 2 4 5 2 4 4 4 4 4 2 4 4 3 3 3 2 5	6 6 9 10 4 7 5 6

Problem K. Snakes On A Plane

Time limit: 0.5 seconds
Memory limit: 256 megabytes

The Cartesian plane is covered with snakes. You will be given a rectangular portion of the plane, divided into a grid of squares. Each square will either contain a segment of a snake or a barrier.

Each snake occupies a chain of adjacent squares, connected horizontally and/or vertically. Snakes are at least 2 segments long (one segment per square) and cannot overlap each other or themselves. Each snake must meet at least one of the following two conditions (see the figures below for examples):

- Both endpoints must be in squares on the edge of the rectangle.
- Both endpoints must be horizontally or vertically adjacent to each other, and the snake is at least 4 segments long (so the snake forms a loop).

The portion of the plane will be given as a grid, where each character represents one square. A `'#'` represents a barrier and a `'.'` (period) represents a square that contains a segment of a snake. Fill the grid with snakes so that all of the non-barrier squares are filled. Do this in such a way that minimizes n — the number of snakes whose endpoints are not adjacent (and therefore, must be on the edge of the grid). Print n, or print `'-1'` if there is no way to fill every non-barrier square.

Input

The first line contains two integers n and m ($1 \le n, m \le 12$) — the sizes of the grid.

The following n lines contain one row per line. Each row contains exactly n characters. Each character is either `'#'` or `'.'`.

Output

Print only one integer — the minimum number of snakes whose endpoints are not adjacent or print `'-1'` if there is no way to fill every non-barrier square.

Examples

standard input	standard output
6 6 ..... .#.##. .#.... ....#. .##.#.	2
5 7 ###.### ###.### ..... ###.### ###.###	-1
7 8 #.....## ...#.... ....#.#.#. .....#. ...#..#. #.....	1

Note

The first example the grid can be filled with snakes in several ways and a few are shown in the figures below:

In each figure, there are 2 snakes that do not form a loop. There is no way to fill the grid with only 1, so 2 is the correct answer for this input.

The third example is shown in the figure below:

Problem L. Colorful Balls

Time limit: 2 seconds
Memory limit: 256 megabytes

We have a bag that contains n balls of different colors. In each turn we pick two balls from the bag, one after another, and paint the second one with the first one's color. After the paint dries, we put both balls back into the bag and shuffle its contents.

You are given the initial colors as a string s. More precisely, each character in the string s corresponds to a single ball. Balls represented by equal characters have the same color.

Print the expected number of turns until all the balls have the same color.

Input

The first line contains the only string s — the initial colors of the balls. That string contains from 1 to 24 characters. Each character is uppercase English letter.

Output

Print only one value — the expected number of turns until all the balls have the same color. The printed value must be accurate to within a relative or absolute value of 10^{-9}.

Examples

standard input	standard output
AB	1
Q	0
AAAAAAA	0
ZCZ	3

Note

In the first example in the first turn we will paint one ball with the other's color, and we are done.

The second example if there is just a single ball, all balls already have the same color, and thus the correct answer is zero.

Problem M. Prime Sums

Time limit: 0.5 seconds
Memory limit: 256 megabytes

The array of integers a describes a bag of non-negative integers. A bag is the same thing as a set, only it may contain repeated elements. The order of elements in a bag does not matter.

Given two bags A and B, we say that A is a sub-bag of B if A can be obtained by erasing zero or more elements from B.

The weight of a bag is the sum of its elements.

Examples:

The bags $(1, 2, 1, 3, 1)$ and $(3, 1, 1, 1, 2)$ are the same, but different from the bag $(1, 2, 3, 3)$.

Bags $(1, 2)$ and $(3, 1, 1)$ are sub-bags of the bag $(1, 2, 1, 3, 1)$, but bag $(1, 2, 2)$ is not.

The weight of the bag $(1, 2, 1, 3, 1)$ is $1 + 2 + 1 + 3 + 1 = 8$.

Print how many sub-bags of bag have a prime weight.

Input

The first line contains only one integer n ($1 \le n \le 50$) — the number of elements in a.

The second line contains n space-separated integers a_i ($0 \le a_i \le 10000$) — the elements of a.

Output

Print only one integer — how many sub-bags of bag have a prime weight.

Examples

standard input	standard output
4 1 1 2 7	5
10 1 1 1 1 1 1 1 1 1 1	4
6 4 6 8 10 12 14	0

Note

A prime number is a positive integer with exactly two positive integer divisors. Zero (0) and one (1) are not prime numbers.

The bag in the first example has 12 different sub-bags: $(1, 1, 2, 7)$, $(1, 2, 7)$, $(2, 7)$, $(1, 1, 7)$, $(1, 7)$, (7), $(1, 1, 2)$, $(1, 2)$, (2), $(1, 1)$, (1), and $()$. Out of these 12 5 have prime weights: $(1, 1, 2, 7)$ has weight 11, (7) has weight 7, $(1, 2)$ has weight 3, and both (2) and $(1, 1)$ have weight 2.

In the second example the bag has eleven different sub-bags. Out of them four have prime weights (2, 3, 5, and 7).

Problem N. Sum And Product

Time limit: 0.5 seconds
Memory limit: 256 megabytes

A list of non-negative numbers is called satisfactory if the sum of the numbers in the list is equal to s and the product of the numbers is equal to p. Find a satisfactory list with the least possible number of elements, and print its size. If no such list exists, print `'-1'` instead. **Please note that the list may contain non-integer numbers.**

Input

The first line contains two integers s and p ($1 \le s, p \le 10^9$).

Output

Print only one integer — the minimal size of a satisfactory list or print `'-1'` if no such list exists.

Examples

standard input	standard output
10 10	1
10 20	2
10 30	3
10 40	-1

Note

The first example the list contains only one element: 10.

The second example the list contains two elements: 2.7639... and 7.2360....

Problem O. Polygons

Time limit: 0.5 seconds
Memory limit: 256 megabytes

You are given n line segments numbered 1 to n. Compose a convex k-sided polygon, where each side is one of the given segments. Each segment can only be used once in the polygon.

Print the number of different polygons you can compose. Two polygons are considered different if there exists a segment i such that one of the polygons contains segment i but the other polygon does not.

Input

The first line contains only one integer n ($1 \le n \le 50$) — the number of line segments.

The second line contains n space-separated integers a_i ($1 \le a_i \le 50000$) — the segments.

The last line contains only one integer k ($3 \le k \le 10$) — the size of the polygon.

Output

Print only one integer — the number of different polygons you can compose.

Examples

standard input	standard output
4 1 1 1 1 3	4
4 2 3 4 5 3	3
6 4 4 4 2 2 2 3	11

Note

A convex polygon can be constructed from a set of segments if the length of each segment from this set is strictly less than the sum of lengths of the remaining segments.

The first example a nondegenerate triangle can be built using any triple from the given segments.

The second example any triple except $2, 3, 5$ will do.

In the third example you can a make nondegenerate triangle using three segments of length 2, or three segments of length 4, or any two segments of length 4 with any segment of length 2.