Problem A. Another Coin Weighing Puzzle

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

You have some bags of coins. Each bag contains exactly k coins. Exactly one bag contains only counterfeit coins (we'll call this the *fake bag*), while all other bags contain only real coins. All real coins weigh exactly the same number of grams. All counterfeit coins weigh exactly the same number of grams. You don't know the exact weights of a real or counterfeit coin. You do know a counterfeit coin is strictly heavier than a real coin, but you do not know exactly how much heavier it is. The weights of the coins are positive real numbers.

You have a scale which you can use at most m times. The scale has a left and right side. To use the scale, you can place any number of coins, taken from any of the bags, on each side of the scale, as long as the total number of coins on the left and right sides are exactly equal. The scale will return a single real number s. If s is zero, both sides of the scale weigh exactly the same. If s is negative, the left side is $|s|$ grams heavier than the right side. If s is positive, the right side is s grams heavier than the left side. Coins can be reused multiple times for different weighings, and you are able to keep track of which bag each coin came from. You must specify beforehand all weighings you want to perform (so you cannot adjust what gets weighed in future trials based on the results of previous trials). After using the scale m times, you would like to be able to determine which bag is the fake bag.

You are now wondering: given m and k, what is the maximum number of bags for which you can always determine the fake bag? This number can get large, so output it modulo the large prime 998 244 353.

Input

The single line of input contains two space-separated integers m and k ($1 \le m, k \le 10^6$), where m is the number of weighings available to you and k is the number of coins in each bag.

Output

Output a single integer, which is the maximum number of bags for which you can determine the fake bag in m weighings, modulo the large prime 998 244 353.

Examples

standard input	standard output
2 1	9
2 2	17

Note

One way we can use 2 weighings to determine the fake bag among 9 bags, each containing 1 coin, is as follows:

- On the first weighing, put the coins from bags 1,2,3 on the left, and the coins from bags 7,8,9 on the right.
- On the second weighing, put the coins from bags 1,4,7 on the left, and the coins from bags 3,6,9 on the right.

We can determine the fake bag as follows:

- The first weighing tells us which group of bags $(1, 2, 3)$, $(4, 5, 6)$, $(7, 8, 9)$ contains the fake bag (e.g. if the left side is heavier, then group $(1, 2, 3)$ contains the fake bag, if both sides are equal, then group $(4, 5, 6)$ contains the fake bag, otherwise group $(7, 8, 9)$ contains the fake bag).

- The second weighing will tell us which group of bags $(1, 4, 7)$, $(2, 5, 8)$, $(3, 6, 9)$ contains the fake bag. The resulting fake bag can be uniquely determined as a result.

Problem B. Mini Battleship

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	512 mebibytes

Battleship is a game played by two players. Each player has their own grid, which is hidden from their opponent. Each player secretly places some ships on their grid. Each ship covers a horizontal or vertical straight line of one or more continguous squares. Ships cannot overlap. All ships are considered distinct, even if they have the same size. The orientation of each ship is not important to the game, only the squares they occupy.

After placing their ships, the players then take turns taking shots at their opponent's ships by calling out a coordinate of their opponent's grid. The opponent must honestly say whether the shot was a hit or a miss. When all of a ship's squares are hit, that ship sinks ("You sunk my battleship!!"). A player loses when all of their ships are sunk.

Bob is playing a game of Mini Battleship against Alice. Regular Battleship is played on a 10×10 grid with 5 ships. Mini Battleship is much smaller, with a grid no larger than 5×5 and possibly fewer than 5 ships.

Bob wonders how many ship placements are possible on Alice's board given what he knows so far. The answer will be 0 if Alice is cheating! (Or, if the game setup isn't possible.)

Input

The first line of input contains two space-separated integers n ($1 \le n \le 5$) and k ($1 \le k \le 5$), which represent a game of Mini Battleship played on an $n \times n$ grid with k ships.

Each of the next n lines contains a string s ($|s| = n$). This is what Bob sees of Alice's grid so far.

- A character '`X`' represents one of Bob's shots that missed.
- A character '`O`' (Letter O, not zero) represents one of Bob's shots that hit.
- A dot ('`.`') represents a square where Bob has not yet taken a shot.

Each of the next k lines contains a single integer x ($1 \le x \le n$). These are the sizes of the ships.

Output

Output a single integer, which is the number of ways the k distinct ships could be placed on Alice's grid and be consistent with what Bob sees.

Examples

standard input	standard output
4 3 .... .OX. .... O..X 3 2 1	132
4 4 .X.X .XX. ...X 1 2 3 4	6
2 2 .. .. 2 2	4

Problem F. Hopscotch

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

There's a new art installation in town, and it inspires you... to play a childish game. The art installation consists of a floor with an $n \times n$ matrix of square tiles. Each tile holds a single number from 1 to k. You want to play hopscotch on it. You want to start on some tile numbered 1, then hop to some tile numbered 2, then 3, and so on, until you reach some tile numbered k. You are a good hopper, so you can hop any required distance. You visit exactly one tile of each number from 1 to k.

What's the shortest possible total distance over a complete game of Hopscotch? Use the Manhattan distance: the distance between the tile at (x_1, y_1) and the tile at (x_2, y_2) is $|x_1 - x_2| + |y_1 - y_2|$.

Input

The first line of input contains two space-separated integers n ($1 \le n \le 50$) and k ($1 \le k \le n^2$), where the art installation consists of an $n \times n$ matrix with tiles having numbers from 1 to k.

Each of the next n lines contains n space-separated integers x ($1 \le x \le k$). This is the art installation.

Output

Output a single integer, which is the total length of the shortest path starting from some 1 tile and ending at some k tile, or -1 if it isn't possible.

Examples

standard input	standard output
10 5 5 1 3 4 2 4 2 1 2 1 4 5 3 4 1 5 3 1 1 4 4 2 4 1 5 4 5 2 4 1 5 2 1 5 5 3 5 2 3 2 5 5 2 3 2 3 1 5 5 5 3 4 2 4 2 2 4 4 2 3 1 5 1 1 2 5 4 1 5 3 2 2 4 1 2 5 1 4 3 5 5 3 2 1 4 3 5 2 3 1 3 4 2 5 2 5 3 4 4 2	5
10 5 5 1 5 4 1 2 2 4 5 2 4 2 1 4 1 1 1 5 2 5 2 2 4 4 4 2 4 5 5 4 2 4 4 5 5 5 2 5 5 2 2 2 4 4 4 5 4 2 4 4 5 2 5 5 4 1 2 4 4 4 4 2 1 2 4 4 1 2 4 5 1 2 1 1 2 4 4 1 4 5 2 1 2 5 5 4 5 2 1 1 1 1 2 4 5 5 5 5 5 5	-1

Problem G. ICPC Camp

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	512 mebibytes

John is a leading organizer of this year's North America ICPC training camp. The camp lasts several days. On each day, there will be a lecture introducing two problems: one classical problem and one creative problem. Each problem can only be introduced once during the entire camp. Every problem has an integer difficulty level.

John knows that the lecture on each day should not be too overwhelming. Therefore, the sum of the difficulties of the two problems in a single day shall not exceed some fixed value. Also, the two problems on each day should be roughly on the same level. Let d's be the absolute difference between the difficulties of the two problems introduced on any given day. The maximum of all of the ds, defined as D, should be as small as possible.

If John chooses problems well and arranges them wisely, what is the smallest D he can achieve for the n days of the ICPC training camp?

Input

The first line of input contains four space-separated integers n, p, q ($1 \le n, p, q \le 2 \cdot 10^5$, $n \le \min(p, q)$) and s ($0 \le s \le 10^9$), where n is the number of days of the camp, p is the number of classical problems, q is the number of creative problems, and s is the maximum sum of difficulties on any given day.

Each of the next p lines contains an integer x ($0 \le x \le 10^9$). These are difficulties of the p classical problems.

Each of the next q lines contains an integer y ($0 \le y \le 10^9$). These are difficulties of the q creative problems.

Output

Output a single integer, which is the smallest D John can achieve, or -1 if there is no way John can select problems for the n training days.

Examples

standard input	standard output
3 4 5 10 3 4 4 9 0 1 5 6 6	2
4 4 4 15 1 5 10 12 1 3 10 14	13
4 4 4 10 1 12 5 10 1 10 3 14	-1

Problem H. Letter Wheels

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

There are three horizontal wheels of letters stacked one on top of the other, all with the same number of columns. All wheels have one letter, either '`A`', '`B`' or '`C`', in each of its columns on the edge of the wheel. You may rotate the wheels to adjust the positions of the letters. In a single *rotation*, you can rotate any single wheel to the right or to the left by one column. The wheels are round, of course, so the first column and last column are adjacent.

You would like to determine whether it is possible to rotate the wheels so that every column has three distinct letters across the three wheels, and if so, determine the minimum number of rotations required.

Input

The input has exactly three lines. Each line has a string s ($2 \le |s| \le 5 \cdot 10^3$) consisting only of upper-case letters '`A`', '`B`' or '`C`', describing the letters of one wheel in their initial positions. All three strings will be of the same length.

Output

Output a single integer, which is the minimum number of rotations required, or -1 if it isn't possible.

Examples

standard input	standard output
ABC ABC ABC	2
ABBBAAAA BBBCCCBB CCCCAAAC	3
AABB BBCC ACAC	-1

Problem J. Lunchtime Name Recall

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

Mia is a newly hired administrative assistant. Though Mia was introduced to everyone in the company on her first day, she is forgetful and struggles to remember people's names. Being too shy to ask for names again, she discovers a way to recall people's names during lunchtime without asking.

Mia orders lunch for all of her colleagues in each of the next several days. On any day, Mia orders burgers for some of her colleagues and salads for the rest of her colleagues. The number of burgers she orders may vary per day. After placing the order, she sends an email to the colleagues who get a burger for lunch, and also to the remaining colleagues about their salads. Mia has an email list with all her colleagues' names. She may choose by name who gets a burger and who gets a salad. Mia can see her colleagues as they are eating their lunch. Thus, by observing who turns out to eat a burger in the office and who eats a salad, she may gain some information to help her uniquely identify the names of her colleagues.

For example, suppose there are three colleagues with names Alice, Danielle, and Jennifer, and Mia can order one burger plus two salads on each day. On the first day, if Mia orders a burger for Alice and salads for Danielle and Jennifer, she can then tell who is Alice by observing who eats a burger. On the second day, Mia may order a burger for Danielle and a salad for Jennifer (and a salad for Alice who is already identified). Consequently she can uniquely identify all three colleagues.

What is the maximum number of colleagues that Mia can uniquely identify in the next few days, if she allocates the burger and salad recipients optimally?

Input

The first line of input contains two space-separated integers n ($2 \le n \le 30$) and m ($1 \le m \le 10$), where Mia has n colleagues and will be ordering lunch for m days.

Each of the next m lines contains a single integer a ($1 \le a < n$), which is the number of burgers Mia orders on that day. The days are listed in order.

Output

Output a single integer, which is the maximum number of Mia's n colleagues that she can uniquely identify after m days.

Examples

standard input	standard output
4 2 2 2	4
16 3 6 8 8	5

Problem K. Rooted Subtrees

Input file:	*standard input*
Output file:	*standard output*
Time limit:	7 seconds
Memory limit:	512 mebibytes

A *tree* is a connected, acyclic, undirected graph with n nodes and $n-1$ edges. There is exactly one path between any pair of nodes. A *rooted tree* is a tree with a particular node selected as the root.

Let T be a tree and T_r be that tree rooted at node r. The *subtree* of u in T_r is the set of all nodes v where the path from r to v contains u (including u itself). In this problem, we denote the set of nodes in the subtree of u in the tree rooted at r as $T_r(u)$.

You are given q queries. Each query consists of two (not necessarily different) nodes, r and p. A set of nodes S is "obtainable" if and only if it can be expressed as the intersection of a subtree in the tree rooted at r and a subtree in the tree rooted at p. Formally, a set S is "obtainable" if and only if there exist nodes u and v where $S = T_r(u) \cap T_p(v)$.

For a given pair of roots, count the number of different non-empty obtainable sets. Two sets are different if and only if there is an element that appears in one, but not the other.

Input

The first line contains two space-separated integers n and q ($1 \le n, q \le 2 \cdot 10^5$), where n is the number of nodes in the tree and q is the number of queries to be answered. The nodes are numbered from 1 to n.

Each of the next $n-1$ lines contains two space-separated integers u and v ($1 \le u, v \le n$, $u \ne v$), indicating an undirected edge between nodes u and v. It is guaranteed that this set of edges forms a valid tree.

Each of the next q lines contains two space-separated integers r and p ($1 \le r, p \le n$), which are the nodes of the roots for the given query.

Output

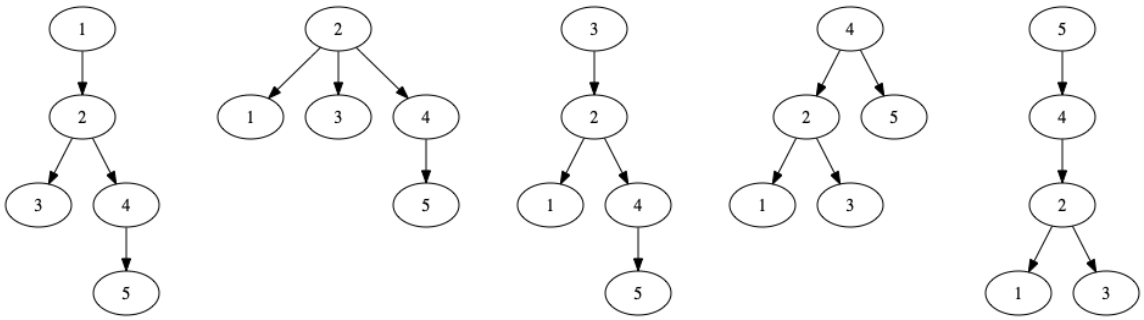
For each query output a single integer, which is the number of distinct obtainable sets of nodes that can be generated by the above procedure.

Example

standard input	standard output
5 2 1 2 2 3 2 4 4 5 1 3 4 5	8 6

Note

The possible rootings of the first tree are



Considering the rootings at 1 and 3, the 8 obtainable sets are:

1. $\{1\}$ by choosing $u = 1$, $v = 1$,
2. $\{1, 2, 4, 5\}$ by choosing $u = 1$, $v = 2$,
3. $\{1, 2, 3, 4, 5\}$ by choosing $u = 1$, $v = 3$,
4. $\{2, 3, 4, 5\}$ by choosing $u = 2$, $v = 3$,
5. $\{2, 4, 5\}$ by choosing $u = 2$, $v = 2$,
6. $\{3\}$ by choosing $u = 3$, $v = 3$,
7. $\{4, 5\}$ by choosing $u = 2$, $v = 4$,
8. and $\{5\}$ by choosing $u = 5$, $v = 5$.

If the rootings are instead at 4 and 5, there are only 6 obtainable sets:

1. $\{1\}$ by choosing $u = 1$, $v = 1$,
2. $\{1, 2, 3\}$ by choosing $u = 2$, $v = 4$,
3. $\{1, 2, 3, 4\}$ by choosing $u = 4$, $v = 4$,
4. $\{1, 2, 3, 4, 5\}$ by choosing $u = 4$, $v = 5$,
5. $\{3\}$ by choosing $u = 3$, $v = 2$,
6. and $\{5\}$ by choosing $u = 5$, $v = 5$.

For some of these, there are other ways to choose u and v to arrive at the same set.

Problem L. Tomb Raider

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

Hu the Tomb Raider has entered a new tomb! It is full of gargoyles, mirrors, and obstacles. There is a door, with treasure beyond. Hu must unlock the door guarding the treasure. On that door is written, in an ancient tongue, the secret to opening the door:

Every face of every gargoyle shall see a face of a gargoyle.

This means that the gargoyles must be rotated in such a way that there is a path for a beam of light to connect each gargoyle's face to another gargoyle's face (possibly its own). The beam of light is reflected by mirrors.

The floorplan of the tomb can be described as a rectangular $n \times m$ grid of cells:

A dot ('`.`') represents an empty cell.
A hash ('`#`') represents an obstacle.
A slash ('`/`') represents a double-sided mirror, as does a Backslash ('\') .
A character '`V`' represents a gargoyle with two faces facing top and bottom.
A character '`H`' represents a gargoyle with two faces facing left and right.

In addition to the '\' and '`/`' mirrors, the tomb is surrounded by walls of mirrors. The following common sense about light is assumed:

1. Light travels in a straight line through empty cells.
2. Two beams of light can intersect without interfering with each other.
3. A '\' mirror reflects light coming from the top/bottom/left/right to the right/left/bottom/top. A '`/`' mirror reflects light coming from the top/bottom/left/right to the left/right/top/bottom.
4. Light is reflected by 180 degrees when it hits a wall (walls are all mirrors).
5. Light is blocked by obstacles and gargoyles.

Hu may rotate any gargoyle by 90 degrees. As time is running short, he wants to know the minimum number of gargoyles that have to be rotated in order to unlock the treasure door.

Input

The first line of input contains two space-separated integers n and m ($1 \le n, m \le 500$), which are the dimensions of the tomb.

Each of the next n lines contains a string s ($|s| = m$) with the characters described above. This is the floorplan of the tomb.

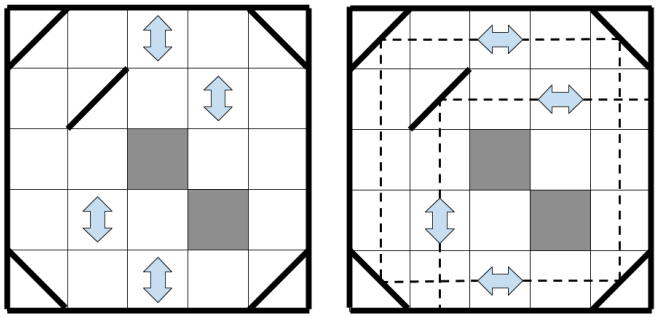
Output

Output a single integer, which is the minimum number of gargoyles that have to be rotated in order to unlock the treasure door. If the puzzle has no solution, output -1.

Examples

standard input	standard output
5 5 /.V.\ ./.V. ..#.. .V.#. \.V./	3
2 5 V...\ H...V	-1
2 2 VV VV	0

Note



The above are illustrations of Sample Input/Output 1 with the initial configuration on the left and the solution of the puzzle on the right. Three gargoyles are rotated to solve the puzzle.

Problem M. Morse Encoding

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

Morse code is an assignment of sequences of dot and dash symbols to alphabet characters. You are to reassign the sequences of Morse code so that it yields the shortest total length to a given message, and return that total length.

A dot symbol has length 1. A dash symbol has length 3. The gap between symbols within a character encoding has length 1. The gap between character encodings has length 3. Spaces, punctuation, and alphabetic case are ignored, so the text "`The quick brown dog jumps over the lazy fox`". is encoded as though it were "`THEQUICKBROWNDOGJUMPSOVERTHELAZYFOX`".

For instance, for the input "`ICPC`, the answer is 17. Encode the '`C`'s with a single dot, the '`I`' with a dash, and the '`P`' with two dots, and have resulting length $(3) + 3 + (1) + 3 + (1 + 1 + 1) + 3 + (1) = 17$.

Input

The single line of input consists of a string s ($1 \le |S| \le 32000$) of upper-case or lower-case letters, spaces, commas, periods, exclamation points, and/or question marks. Everything but the letters should be ignored. The line will contain at least one letter.

Output

Output a single integer, which is the length of s when encoded with an optimal reassignment of the sequences of Morse code.

Example

standard input	standard output
ICPC	17
A	1

Problem N. Checkerboard

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

An $r \times c$ grid of squares is to be colored in a checkerboard style. The board will be filled with rectangles made up of the grid squares. The heights and widths of the rectangles will be specified. Black and White are the only two colors of the rectangles. Any two adjacent rectangles that share a side should be colored differently. The top-left rectangle should be Black. Print the checkerboard.

Input

The first line contains four space-separated integers r, c, v and h ($1 \le v \le r \le 50$, $1 \le h \le c \le 50$) where the checkerboard is to have r rows and c columns, with v rectangles vertically and h rectangles horizontally.

Each of the next v lines contain a single positive integer a. The sum of the a values will be exactly r. These are the heights of the v rectangles in each column, in order from top to bottom.

Each of the next h lines contain a single positive integer b. The sum of the b values will be exactly c. These are the widths of the h rectangles in each row, in order from left to right.

Output

Print the described checkerboard, in the form of r strings of length c, one per line. The strings should only contain the characters upper-case 'B' (for a Black square) and upper-case 'W' (for a White square).

Examples

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

Problem O. Even or Odd

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

Your friend has secretly picked n consecutive positive integers between 1 and 10^{18} and wants you to guess if their sum is even or odd.

If the sum must be even, write 2. If the sum must be odd, write 1. If the sum could be even or could be odd, write 0.

Input

The single line of input contains a single integer n ($1 \le n \le 10^9$).

Output

Output 2 if the sum of any n consecutive integers in the range from 1 to 10^{18} must be even, 1 if the sum must be odd, or 0 if the sum could be either even or odd.

Example

standard input	standard output
3	0
6	1
12	2

Problem P. Levenshtein Distance

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

The Levenshtein Distance between two strings is the smallest number of simple one-letter operations needed to change one string to the other. The operations are:

- Adding a letter anywhere in the string.
- Removing a letter from anywhere in the string.
- Changing any letter in the string to any other letter.

Given a specific alphabet and a particular query string, find all other unique strings from that alphabet that are at a Levenshtein Distance of 1 from the given string, and list them in alphabetical order, with no duplicates.

Note that the query string must not be in the list. Its Levenshtein Distance from itself is 0, not 1.

Input

Input consists of exactly two lines. The first line of input contains a sequence of unique lowercase letters, in alphabetical order, with no spaces between them. This is the alphabet to use. The second line contains a string s ($2 \le |s| \le 100$), which consists only of lower-case letters from the given alphabet. This is the query string.

Output

Output a list, in alphabetical order, of all strings which are a Levenshtein Distance of 1 from the query string s. Output one word per line, with no duplicates.

Examples

standard input	standard output
eg egg	eeg eegg eg ege egeg egge eggg gegg gg ggg