Problem C. Magic Checkerboard

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

Consider an $n \times m$ checkerboard. On each cell of the checkerboard, place a positive integer. The values in each column of the checkerboard must be in strictly increasing order from top to bottom, and the values in each row of the checkerboard must be in strictly increasing order from left to right.

```
1 2 3 4
3 4 5 6
5 6 7 8
7 8 9 10
```

A Magic Checkerboard has an additional constraint. The cells that share only a corner must have numbers of different parity (Even vs Odd). Note that the following checkboard is invalid, because 2 and 4 share only a corner and have the same parity:

```
1 4
2 6
```

The first 4×4 example is a valid Magic Checkboard. Given a partially filled magic checkboard, can you fill the remaining locations on the checkboard, so that the sum of all values is as small as possible?

Input

Input starts with a line with two space-separated integers n and m ($1 \le n, m \le 2000$), representing the number of rows (n) and the number of columns (m) of the checkerboard. Each of the next n lines will contain m space-separated integers c ($0 \le c \le 2000$), representing the contents of the checkerboard. Zero is used for cells without numbers that you must fill in. You may use any positive integers to fill in the cells without numbers, so long as you form a valid Magic Checkerboard. You are not limited to numbers not less than 2000, and the numbers are not required to be unique.

Output

Output a single integer representing the minimum sum possible by replacing the 0-cells with positive integers to form a valid Magic Checkerboard. Output -1 if it is not possible to replace the 0-cells to meet the constraints of a Magic Checkerboard.

Examples

standard input	standard output
4 4 1 2 3 0 0 0 5 6 0 0 7 8 7 0 0 10	88
4 4 1 2 3 0 0 0 5 6 0 4 7 8 7 0 0 10	-1
2 3 0 0 0 0 0 0	18

Problem E. Primal Partitions

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

The Math department has challenged the Computer Science department at your University to solve a difficult puzzle in Discrete Mathematics. With the pride of your Computer Science department at stake, you must solve this puzzle!

In this puzzle, you are given a sequence of n positive integers. The sequence must be partitioned into k consecutive contiguous regions, with at least 1 integer in each region. After finding a partition, it is scored in a strange way. In each region, you must find the largest prime number that divides every number in that region. The Math department reminds you a prime number is an integer greater than 1 where its only divisors are 1 and itself. If you cannot find such a prime, your score for that region is 0. Your total score for the partition is the minimum over all regions.

Your task is to find the maximum possible score. The Math department will win if their score is better, so be sure to find the maximum each time!

Input

Input begins with a line with two space-separated integers n ($1 \leq n \leq 2 \cdot 10^4$) and k ($1 \leq k \leq min(100, n)$), where n is the number of positive integers in the original sequence, and k is the number of regions in the partition. The next line contains n space-separated integers v ($1 \leq v \leq 10^6$). This is the sequence to be partitioned, in order.

Output

Output a single integer representing the maximum score possible partitioning the sequence of n positive integers into k regions.

Examples

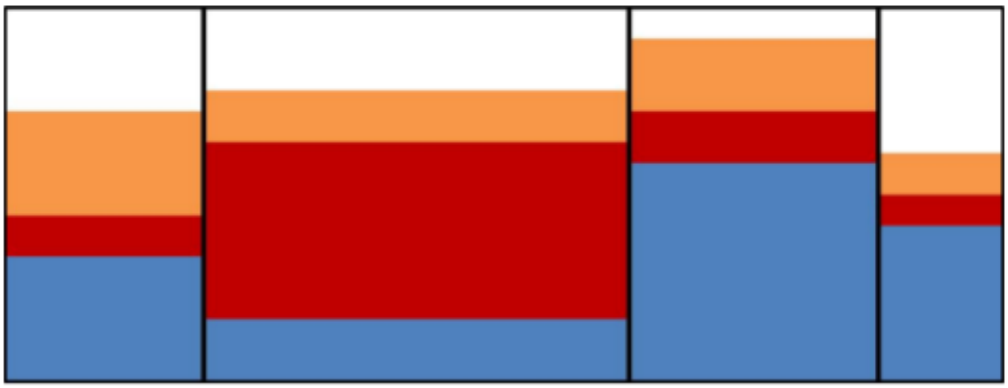
standard input	standard output
5 3 10 5 4 8 3	2
5 3 10 11 12 13 14	0

Problem F. Sand Art

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	512 mebibytes

At art shows, it is very common to have booths where children can create their very own sand art. This art is typically made by taking a jar or bottle and filling it with layers of different colors of sand. Instead of a bottle, this year a new container is being used for decorating! The container is a glass box!

The box has a 2D rectangular face and a thickness of exactly 1 unit. Inside the glass box, $n-1$ vertical dividers are placed to separate it into n sections. In the example below, the box is divided into 4 sections using 3 dividers.



Sometimes the children want certain amounts of each color to be in each section of their work of art. They specify a minimum and maximum for each combination of section and sand color. Your task is to help them find how balanced they can make the artwork. This is done by minimizing the difference between the sand heights in the section with the highest overall sand level and the section with the lowest overall sand level.

Input

Input begins with a single line with 4 space-separated integers, n, m, w, h, where:

- n ($2 \le n \le 200$) is the number of sections;
- m ($1 \le m \le 200$) is the number of colors of sand;
- w, h ($1 \le w, h \le 5000$) are the width and height of the box (it always has a depth of 1)

The next line will contain m space-separated real numbers (with at most 3 decimal places) v ($0 < v \le w \cdot h$), which represent the volumes of each color of sand. It is not necessary to use all of the sand, but the minimums for each section must be satisfied.

The next line will have $n-1$ space-separated real numbers with at most 3 decimal places) x ($0 < x < w$) which represent the distance from the left wall of each divider. The x's are guaranteed to be sorted in increasing order.

The next n lines will each have m space-separated real numbers (with at most 3 decimal places) min ($0 \le min \le w \cdot h$). The j-th element of the i-th row is the minimum amount of sand color j to be put in section i.

The next n lines will each have m space-separated real numbers (with at most 3 decimal places) max ($0 \le max \le w \cdot h$). The j-th element of the i-th row is the maximum amount of sand color j to be put in section i, and $min_{ij} \le max_{ij}$.

Output

Output a real number with absolute error 10^{-3} or less representing the minimum difference possible between the maximum and minimum heights of sand in the sections. A distribution of sand will always be possible that satisfies the constraints in the input.

Examples

standard input	standard output
2 2 5 5 2.0 2.0 4.0 1.0 0.0 0.0 1.0 1.0 0.0 0.0 2.0	0.75
2 2 5 5 2.0 2.0 4.0 1.0 0.0 0.0 1.0 1.5 0.0 0.0 2.0	0.625
2 5 11 10 3.0 4.0 4.0 9.0 2.0 4.0 2.0 2.0 1.0 0.5 0.25 0.0 2.0 0.0 4.0 1.0 2.0 2.0 3.0 4.0 0.75 0.0 2.1 0.0 5.1 1.1	0.266

Problem H. Vending Machine

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

Sleazy Bob has happened upon a vending machine. After watching enough people buy tasty snacks, Bob has realized that the vending machine is broken!

Here's what Sleazy Bob observes:

1. A person tries to buy a snack
2. The vending machine then checks to see if there are any left of that snack
3. If there are any left, the machine charges the person for that snack
4. If the machine successfully charges the person, it then gives the person a different snack! Possibly no snack at all, if the machine is out of the different snack!

Sleazy Bob notices that, although the machine is broken, it is at least consistent. Whenever a customer buys a snack from position i, the machine vends from position $f(i)$, where f is some predictable function.

Now, Bob wants to make a profit from the machine. He wants to buy some snacks from the machine, and then turn around and sell the snacks he gets for the market price of the snack. This may be different from the vending price. If a cheap snack is at i, and an expensive snack is at $f(i)$, he could rake in the cash!

Assuming Bob can always find buyers for his snacks, what is the maximum net gain that Bob can obtain by buying some number, possibly zero, of snacks and turning around and selling them later? You may assume that Bob has enough money from other shady activities to cover the cost of buying any number of snacks from the vending machine.

Input

Input begins with a line with a single integer n ($1 \le n \le 10^5$), which is the number of snack positions in the machine. Each of the next n lines contains 4 space-separated integers, f, p,m and s, which describe a snack position in the machine, in order from 1 to n, where:

- f ($1 \le f \le n$) is the value of $f(i)$. That is, it is the position from which the machine will vend when Bob buys from this position;
- p ($1 \le p \le 10^6$) is the price Bob must pay to buy from this position;
- m ($1 \le m \le 10^6$) is the market price of the snack at this position;
- s ($1 \le s \le 10^6$) is the number of snacks at this position.

Output

Output a single line with a single integer, indicating the maximum profit Sleazy Bob can get from his nefarious abuse of the broken vending machine.

Examples

standard input	standard output
3 1 2 3 1 2 3 4 1 3 4 5 1	3
3 2 2 3 8 3 1 5 6 1 9 4 7	39
5 5 9 2 2 1 1 7 4 2 3 6 3 2 2 9 6 1 4 5 1	22

Problem I. Rainbow Zamboni

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

Pacman is cleaning his ice rink! The rink, like everything in PacMan's universe, wraps! In other words, if you go off the edge of the rectangular rink to the right, you'll end up in the same row on the left side of the rink. Go off of the left side of the rink, and you'll end up on the right. Going up off the top of the rink causes you to end up in the same column but on the bottom. Go off of the bottom, and you'll end up at the top.

What makes this zamboni special is that when it cleans the ice, it leaves a color on the ice! At the start, the ice is colorless (white). Whenever the zamboni is on a cell of ice, it overwrites the color on that cell. Each color is represented by a capital letter of the alphabet. The colors are in alphabetical order. The zamboni switches to the next letter in the alphabet at each step, and wraps at the end of the alphabet. If the current color is P, the next color is Q. If the current color is Z, the next color is A. PacMan uses a very specific algorithm to clean the ice:

stepSize = 1

loop **numSteps** times

Move *stepSize* steps in the current direction

Rotate the direction of the zamboni 90 degrees clockwise

Switch to the next color

$stepSize = stepSize + 1$

end loop

The zamboni is originally facing up. Knowing where the zamboni starts, the size of the rink, and the number of steps taken, determine what the ice looks like after the zamboni has completed its run.

Input

Input consists of a single line with 5 space-separated integers, r, c, i, j, n, where:

- r ($1 \le r \le 2000$) is the number of rows of the ice rink;
- c ($1 \le c \le 2000$) is the number of columns of the ice rink;
- i ($1 \le i \le r$) is the starting row of the zamboni;
- j ($1 \le j \le c$) is the starting column of the zamboni;
- n ($1 \le n \le 10^{18}$) is the number of steps (numSteps in PacMan's algorithm).

Output

Output the state of the ice rink after the zamboni has completed its run, as a $r \times c$ grid of characters. Each character should be a capital letter ('`A`'- '`Z`'), representing the color of the ice, or a dot ('`.`') if the ice is white, or an "at" sign ('`@`') if the cell is the final location of the Zamboni.

Examples

standard input	standard output
5 5 3 3 4	....BBC ..A.C ....C @DDDD
5 5 3 3 7	EG... E@BBC EGA.C EG..C FGFFF

Problem J. Zig Zag Nametag

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

When ninjas go to conferences they wear fake nametags. One ninja in particular wants to impress his Sensei. His Sensei chooses a new favorite number every day. The pupil wants to put a name on his nametag that encodes his Sensei's favorite number! This name will consist of only lower case letters. He assigns a value to each letter, based on its position in the alphabet (e.g. $a = 1$, $b = 2$, ..., $z = 26$). Then, he encodes the Sensei's number by adding up the absolute values of the differences of every consecutive pair of letters. For example, the string "`azxb`" has the value of:

$$|a - z| + |z - x| + |x - b| = |1 - 26| + |26 - 24| + |24 - 2| = 49$$

The name that the ninja will write on his nametag is the shortest string that encodes to his Sensei's favorite number. If there's more than one string of the shortest length, he'll choose the one that comes first alphabetically. Given the Senseis favorite number, k, find the string that the ninja should put on his nametag.

Input

Input consists of a single line with a single integer k ($1 \le k \le 10^6$), which is the Sensei's favorite number. There will always be a name that encodes to the Sensei's number.

Output

Output a single line with a string of lower case letters, which is the name that the ninja should put on the nametag to impress the Sensei.

Examples

standard input	standard output
1	ab
19	at
77	aoazb

Problem K. Knight Jumps

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

In Chess, a knight can move two squares in one direction and then one in a perpendicular direction. It can 'jump', meaning that it only requires that the destination square be open — the path between can be occupied.

Given a grid, a starting point and destination point, determine the least number of moves the knight must make to get from the start to the destination. Some squares of the grid may be occupied, so that the knight cannot move there.

Input

Each input will begin with two integers n and m ($2 \le n, m \le 100$), indicating the height and width of the grid. Each of the next n lines will hold m characters, representing the grid. The grid will consist only of '`.`' (open square), '`#`' (occupied square), '`K`' (the knight's starting position) or '`X`' (the knight's destination). There will be exactly one '`K`' and exactly one '`X`' in each input.

Output

Print a single line with a single integer indicating the minimum number of moves the knight needs to get to the destination, or -1 if the knight cannot make it.

Examples

standard input	standard output
5 4 K... .... .#.. .#.. ...X	5
3 3 K.. .X. ...	-1

Problem L. Lost Digit

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

You are helping an archaeologist decipher some runical digits. He knows that this ancient society used a Base 10 system, and that they never start a number with a leading zero unless it is exactly zero (single digit). He's figured out most of the digits, as well as a few operators, but he needs your help to figure out the rest.

The professor will give you a simple math expression. He has converted all of the runes he knows into digits. The only operators he knows are addition (+), subtraction (-), and multiplication (*), so those are the only ones that will appear. Each number will be in the range from −999999 to 999999, and will consist of only the digits '0'-'9', possibly a leading '-', and possibly a few ?'s. The ?'s represent a digit that the professor doesn't know (never an operator, an '=', or a leading '-'). All of the ?'s in an expression will represent the same digit, and it won't be one of the other given digits in the expression.

Given an expression, figure out the value of the rune represented by the question mark. If more than one digit works, give the lowest one. If no digit works, well, that's bad news for the professor — it means that he's gotten some of his runes wrong. Output −1 in that case.

Input

Input will consist of a single line, of the form: $< number >< op >< number >=< number >$. Each $< number >$ will consist of only the digits '0'-'9' , with possibly a single leading minus ('-'), and possibly some ?'s. No number will begin with a leading 0 unless it is 0, no number will begin with -0, and no number will have more than 6 places (digits or ?'s). The $< op >$ will separate the first and second $< number >$'s, and will be one of: '+', '-' or '*'. The '=' will always be present between the second and third $< number >$'s. There will be no spaces, tabs, or other characters. There is guaranteed to be at least one ? in every equation.

Output

Print a single line with the lowest digit that will make the equation work when substituded for the ?'s, or print −1 if no digit will work.

Examples

standard input	standard output
1+1=?	2
123*45?=5?088	6
-5?*-1=5?	0
19--45=5?	-1
??*??=302?	5

Problem M. Multiple Top N

Input file:	*standard input*
Output file:	*standard output*
Time limit:	6 seconds
Memory limit:	512 mebibytes

In College Football, many different sources create a list of the Top 25 (or, Top n) teams in the country. Since it's subjective, these lists often differ, but they're usually very similar. Your job is to compare two of these lists, and determine where they are similar.

In particular, you are to partition them into sets, where each set represents the same contiguous positions in both lists, and has the same teams, and is as small as possible. If the lists agree completely, you'll have n sets, where n is the number of teams in each list.

Input

Input will begin with an integer n ($1 \le n \le 10^6$), indicating the number of teams ranked. The next n lines will hold the first list, in order. The team names will appear one per line, and consist of between 1 and 8 capital letters only. After this will be n lines, in the same format, indicating the second list. Both lists will contain the same team names, and all n team names will be unique.

Output

Output the size of each set, in order, one per line.

Examples

standard input	standard output
5 A B C D E A C D B E	1 3 1
3 FIRST SECOND THIRD FIRST SECOND THIRD	1 1 1
3 UNITED CITY CHELSEA CHELSEA UNITED CITY	3

Problem N. New Contest Director

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

Elections of the new Contest Director are hold! Help the election commission by counting votes and telling them the winner. If more than one candidate ties with the most votes, print out all of their names in alphabetical order.

Input

Input will begin with an integer n ($1 \le n \le 1000$), indicating the number of votes. The next n lines will hold the votes. The candidate's names will appear one per line, and consist of between 1 and 20 capital letters only.

Output

Print the name of the candidate with the most votes. If there is a tie, print out all of the names of candidates with the most votes, one per line, in alphabetical order.

Examples

standard input	standard output
5 BILL MIKE BILL BILL MIKE	BILL
5 BILL MARSHA BILL MARSHA ANONYMOUS	BILL MARSHA