Problem A. Tetris alphabet

Input file:	`alphabet.in`
Output file:	`alphabet.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

The game of Tetris is played with four-connected blocks falling down the well having N rows and 20 columns. Each figure is marked with a unique English letter from "A" to "Z".

Your program must, given the state of the well, determine the order in which the blocks fell down.

Input

The first line of input file contains integer N — number of rows. Following N lines contain 20 characters each, with characters that are either a letter from "A" to "Z" or the dot character (ASCII 46), representing an empty cell.

$0 \leqslant N \leqslant 50$.

Output

Output file must contain a string of letters indicating the order in which figures fell down. If there is more than one order, lexicographically smallest one must be printed. Input data will guarantee that at least one nonempty order exists.

Example

alphabet.in	alphabet.out
6 ....XX..... ....MMMM..... ....K..... ....KKK..... .....ZAAA.FFF..... .....ZZZA..F.B.....	BFZAKMX

Problem B. Circular Area

Input file:	`area.in`
Output file:	`area.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Your task is to write a program, which, given two circles, calculates the area of their intersection with the accuracy of two digits after decimal point.

Input

In the single line of input file there are space-separated real numbers $x_1 y_1 r_1 x_2 y_2 r_2$.

They represent center coordinates and radii of two circles.

Output

The output file must contain single real number — the area.

Example

area.in	area.out
20.0 30.0 15.0 40.0 30.0 30.0	608.37

Problem C. Binary Witch

Input file:	`binary.in`
Output file:	`binary.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Once upon a time in the silent depths of digital forests there lived a Binary Witch. She was able to forecast weather, telling for any day in the future whether it will be rainy or sunny.

Her magic was based on the following ancient rule: let $a_1, a_2, \ldots, a_N$ be the sequence of binary digits, where $a_i = 0$ indicates that i-th day was rainy, and $a_i = 1$ — that it was sunny. To predict the weather in day $N + 1$, consider the t-postfix $a_{N-t+1}, a_{N-t+2}, \ldots, a_N$ consisting of the last t elements. If that postfix is encountered somewhere before the position $N - t + 1$, i.e. if there is such $k \leqslant N - t$, that $a_k = a_{N-t+1}, a_{k+1} = a_{N-t+2}, \ldots, a_{k+t-1} = a_N$ then the predicted value will be a_{k+t}.

If there is more than one occurrence of t-postfix, then the rightmost one (with maximal k) will be taken. So, to make a prediction, she tried t-postfixes, consequently for $t = 13, 12, \ldots, 1$, stopping after the first prediction. If neither postfix was found, she predicted rain (0). If prediction for more than one day is needed, it is assumed that all previous days are predicted correctly, so if first predicted value is b, then we make forecast for day $N + 2$ based on $N + 1$ values, where $a_{N+1} = b$.

Because the witch was burned long ago, your task is to write a program to perform her arcane job.

Input

First line of input file contains two integers N ($1 \leqslant N \leqslant 1\,000\,000$) and L ($1 \leqslant L \leqslant 1\,000$), separated by space. Second line contains a string of N characters "0" and "1".

Output

Output file must contain a single string of L characters, which are forecasts for days $N+1, N+2, \ldots, N+L$.

Example

binary.in	binary.out
10 7 1101110010	0100100

Problem D. Burned Calendar

Input file:	calendar.in
Output file:	calendar.out
Time limit:	2 seconds
Memory limit:	256 Mebibytes

A year calendar is printed using the monospace font according to the following rules:

1. All spaces on the printed calendar are represented by the dot character (ASCII 46).
2. Every month occupies a rectangle of 17 by 8 characters, with the name of the month written in all capital letters starting from the 2nd character of the first line.
3. All days of the months are printed in 4, 5, or 6 columns 2 characters wide and 7 characters high, with one space between the columns. The first day of the week is Monday.
4. Months of the year are arranged in the three rows separated by horizontal and vertical lines of spaces. Each row contains four months. The calendar margins are of 1 space from all sides. Therefore, the whole calendar has size of 73 by 28 characters.

Note that January 1st, 1900 was Monday. Also note that a leap year number is divisible by 4 and not divisible by 100, or divisible by 400. For example, a part of the printed calendar from October to December 2002 may look like this:

```
.OCTOBER...........NOVEMBER.........DECEMBER.........
....7.14.21.28........4.11.18.25........2..9.16.23.30.
.1..8.15.22.29........5.12.19.26........3.10.17.24.31.
.2..9.16.23.30........6.13.20.27........4.11.18.25....
.3.10.17.24.31........7.14.21.28........5.12.19.26....
.4.11.18.25........1..8.15.22.29........6.13.20.27....
.5.12.19.26........2..9.16.23.30........7.14.21.28....
.6.13.20.27........3.10.17.24........1..8.15.22.29....
....................................................
```

A calendar was printed and then burned, with only a small rectangular piece left. Your program must determine to which of years from 1900 to 2100 this piece could belong.

Input

The first line of the input file contains integer numbers N and M, separated by spaces — the size of the piece. The following M lines contain N characters each — the piece of calendar.

$2 \leqslant N, M \leqslant 10$.

Output

Output file must contain an ordered list of year numbers, one year per line. If given piece could not belong to any calendar, output must contain a single integer 0 (zero).

Examples

calendar.in	calendar.out
4 4 1..8 .... JUNE 1..8	1903 1914 1925 1931 1942 1953 1959 1970 1981 1987 1998 2009 2015 2026 2037 2043 2054 2065 2071 2082 2093 2099
3 2 1.7 ...	0

Problem E. Falling Cards

Input file:	cards.in
Output file:	cards.out
Time limit:	2 seconds
Memory limit:	256 Mebibytes

It is hard to make the playing card stand on its edge, however, some patient person managed to put N of them upon the desk in such a way that each cards stands on its shorter edge. The top-down view of that desk with cards upon it can be represented with the set of line segments with coordinates (x_i, y_i) to (v_i, w_i). The segments do not intersect.

The first card falls flat on its side, causing any card it touches to fall down also. The angle between vector $(x_1, y1_)$ − (v_1, w_1) and the falling direction of the first card is equal to 90 degrees (measured counter-clockwise). If card A touches card B, the direction of B's fall is chosen so that the continuation of the direction vector does not cross the line containing segment A. Input data are guaranteed to never contain:

1. a card falling exactly perpendicular to the other and
2. a falling card that touches more than one of still standing cards.

Your program must determine which cards will fall, and which shall remain standing.

Input

The first line of input file contains numbers N and H — the number of cards and the floating point height of a card. Each of the following N lines contains four floating-point numbers $x_i y_i v_i w_i$ — coordinates of cards separated by spaces.

$1 \leqslant N \leqslant 100$.

Output

The output file must contain the list of fallen cards' numbers, sorted in increasing order and separated by spaces.

Example

cards.in	cards.out
3 100 10 10 50 40 10 0 50 30 20 90 20 20	1 3

Problem F. Fool Game

Input file:	fool.in
Output file:	fool.out
Time limit:	2 seconds
Memory limit:	256 Mebibytes

The game of "fool" is played with a small set of cards, which includes nine ranks — 6, 7, 8, 9, 0 (10), J (Jack), Q (Queen), K (King), A (Ace) of the four suits each — h (Hearts), s (Spades), d (Diamonds) and c (Crosses). So, for example a queen of spades is denoted Qs, and a ten of diamonds is 0d. One of the suits is declared a *trump*. During the game, the card *beats* another card, if either it has the same suit and higher rank, or it is a trump, while the beaten card is not.

In the game, a move proceeds as following. First player puts one of his cards on the desk, and the second player can either beat it with one of his cards, putting his card over it, or take it, if he has no suitable card. If the card is beaten, first player may flip in any of his remaining cards, which has same rank as any card already on the desk. This card, in turn, may be beaten or taken together with all other cards on the desk, etc. For example, if first player has cards 6s6dQhKd, the second player has 6h7h0sQd and hearts are the trumps, then first player can move with Kd, which is beaten with 6h, then flip in 6s, beaten by 0s, then 6d, beaten by Qd and at last Qh, which can not be beaten with 7h, so the second player has to take it.

Your task is to write a program that, given the trump suit and first and second player's cards, determines for the first player such a move as to eventually make the second player take.

If there is more than one such move, the program must find one with smallest rank. If there is several moves with smallest rank, program must choose the card with the first suit in the order mentioned in the first paragraph (i. e. $h < s < d < c$). In the example above, second player could beat Kd with 7h, thus preventing further flips. On the other hand, move of Qh will be immediately taken.

Input

In the first line of input file there is a single character h, s, d, or c, determining a trump suit. On the second line there is a string denoting first player's cards, and on the third line — the string with the second player's cards. All cards are different, and the players have *equal number* of cards.

Output

Output file must contain a single line with either a card to move or a string "NO", if a program was unable to find it.

Example

fool.in	fool.out
s As7h8dJc Jd8s7c0c	7h

Problem G. Hamming Problem

Input file:	hamming.in
Output file:	hamming.out
Time limit:	2 seconds
Memory limit:	256 Mebibytes

For each three prime numbers p_1, p_2 and p_3, let's define Hamming sequence $H_i(p_1, p_2, p_3)$, $i = 1, \ldots$ as containing in increasing order all the natural numbers whose only prime divisors are p_1, p_2 or p_3.

For example, $H(2, 3, 5) = 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, 18, 20, 24, 25, 27, \ldots$ So $H_5(2, 3, 5) = 6$.

Input

In the single line of input file there are space-separated integers p_1 p_2 p_3 i. All numbers in input are less than 10^{18}.

Output

The output file must contain the single integer — $H_i(p_1, p_2, p_3)$. All numbers in output are less than 10^{18}.

Example

hamming.in	hamming.out
7 13 19 100	26590291

Problem H. Longest Ordered Subsequence

Input file:	longest.in
Output file:	longest.out
Time limit:	2 seconds
Memory limit:	256 Mebibytes

A numeric sequence of a_i is ordered if $a_1 < a_2 < \ldots < a_N$. Let the subsequence of the given numeric sequence $(a_1, a_2, \ldots, a_N)$ be any sequence $(a_{i_1}, ai_2, \ldots, a_{i_K})$, where $1 \leqslant i_1 < i_2 < \ldots < i_K \leqslant N$. For

example, the sequence $(1, 7, 3, 5, 9, 4, 8)$ has ordered subsequences, e. g., $(1, 7)$, $(3, 4, 8)$ and many others. All longest ordered subsequences of this sequence are of length 4, e. g., $(1, 3, 5, 8)$.

Your program, when given the numeric sequence, must find the length of its longest ordered subsequence.

Input

The first line of input file contains the length of sequence N. The second line contains the elements of sequence — N integers in the range from 0 to 10 000 each, separated by spaces.

$1 \leqslant N \leqslant 1000$.

Output

Output file must contain a single integer — the length of the longest ordered subsequence of the given sequence.

Example

longest.in	longest.out
7 1 7 3 5 9 4 8	4

Problem I. Network Saboteur

Input file:	network.in
Output file:	network.out
Time limit:	2 seconds
Memory limit:	256 Mebibytes

A university network is composed of N computers. System administrators gathered information on the traffic between nodes, and carefully divided the network into two subnetworks in order to minimize traffic between parts.

A disgruntled computer science student Vasya, after being expelled from the university, decided to have his revenge. He hacked into the university network and decided to reassign computers to maximize the traffic between two subnetworks.

Unfortunately, he found that calculating such worst subdivision is one of those problems he, being a student, failed to solve. So he asks you, a more successful CS student, to help him.

The traffic data are given in the form of matrix C, where C_{ij} is the amount of data sent between i-th and j-th nodes ($C_{ij} = C_{ji}, C_{ii} = 0$). The goal is to divide the network nodes into the two disjointed subsets A and B so as to maximize the sum of all C_{ij}, where i belongs to A, and j belongs to B.

Input

The first line of input file contains a number of nodes N. The following N lines, containing N space-separated integers each, represent the traffic matrix C.

$2 \leqslant N \leqslant 20$; $0 \leqslant C_{ij} \leqslant 10\,000$.

Output

Output file must contain a single integer — the maximum traffic between the subnetworks.

Example

network.in	network.out
3 0 50 30 50 0 40 30 40 0	90

Problem J. PreQueL

Input file:	prequel.in
Output file:	prequel.out
Time limit:	2 seconds
Memory limit:	256 Mebibytes

In some simplistic DBMS named PreQueL, the only column type allowed is CHAR(1) (a single character), and furthermore, its values are restricted to English upper-case letters (A to Z). Table may contain up to 9 columns, numbered from 1 to 9. Tables themselves are named with lower-case English letters (a to z).

The only database query possible first joins all the tables, then selects some rows according to conditions in one of two forms: either `<column>=<value>` or `<column1>=<column2>`, for example `a2=A` or `b1=c4`. All conditions must hold simultaneously, as if they were connected by AND operator.

You must write a PreQueL processor, which, given a tables and a set of conditions, will produce query result, i.e. those rows of a join satisfying all the conditions. Resulting rows must be sorted alphabetically.

Input

The first line of input file contains of two integers — number of tables T and number of conditions D.

Starting from the second line there are T tables represented with number of rows RN and number of columns CN in the first line of a table, which is followed by RN lines consisting of exactly CN characters each. D lines with conditions follow the whole set of tables.

The constraints are: $1 \leqslant T \leqslant 26$, $1 \leqslant D \leqslant 50$, $1 \leqslant C \leqslant 9$, $1 \leqslant R \leqslant 1000$.

Output

Output file contains result rows, one row per line. No input query will produce more than 1000 rows.

Example

prequel.in	prequel.out
2 2 3 2 AX BX BY 2 3 ACD BCC a1=b1 a2=X	AXACD BXBCC

Problem K. Very Simple Problem

Input file:	`simple.in`
Output file:	`simple.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

During a preparation of programming contest, its jury is usually faced with many difficult tasks. One of them is to select a problem simple enough to most, if not all, contestants to solve.

The difficulty here lies in diverse meanings of the term "simple" amongst the jury members. So, the jury uses the following procedure to reach a consensus: each member weights each proposed problem with a positive integer "complexity rating" (not necessarily different for different problems). The jury member calls "simplest" those problems that he gave the minimum complexity rating, and "hardest" those problems that he gave the maximum complexity rating.

The ratings received from all jury members are then compared, and a problem is declared as "very simple", if it was called as "simplest" by more than a half of the jury, and was called as "hardest" by nobody.

Input

The first line of input file contains integers N and P, the number of jury members and the number of problems. The following N lines contain P integers in range from 0 to 1 000 each — the complexity ratings.

$1 \leqslant N, P \leqslant 100$.

Output

Output file must contain an ordered list of "very simple" problem numbers, separated by spaces. If there are no such problems, output must contain a single integer 0 (zero).

Example

simple.in	simple.out
4 4 1 1 1 2 5 900 21 40 10 10 9 10 3 4 3 5	3

Problem L. Spreadsheet

Input file:	`spreadsheet.in`
Output file:	`spreadsheet.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

You are to write a program emulating a very simple spreadsheet application. It works with a table with 9 rows, from 1 to 9, and 26 columns, from A to Z. Table cells are referenced by names composed of column and row codes, ex. $B1$, $S8$.

Each cell contains an expression up to 255 characters long. Expressions use integer constants, cell references, parenthesis, operators `+`, `-`, `*`, and `/` (whole division). For example: `567`, `E8/2`, `(3+B3)*(C4-1)` are all valid expressions. All operators are whole, all arguments and results are guaranteed to be less than 1 000 000. Division by zero yields zero.

If the value of the cell referenced by some expression is not defined, it is presumed to be 0 (zero). The situation when two or more cells are mutually dependent on each other is considered a special case of *circular reference*.

Input

First input line contains number of expressions N. Following N lines are in format `<Cell reference>=<expression>`. All expressions are correct, and each cell is defined by at most one expression.

Output

Output file must contain single line with either the value of the cell `A1` or number 1 000 000 (one million) if the value of `A1` cannot be computed due to a circular reference.

Example

spreadsheet.in	spreadsheet.out
4 A1=B1+C5 B1=20 C5 =B1 /D7-E1*E1 E1=(3+1)*2	-44