Problem A. Balance

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

We say that a matrix A of size $N \times N$ is *balanced* if $A[i][j] + A[i+1][j+1] = A[i+1][j] + A[i][j+1]$ for all $1 \le i, j \le N-1$.

You are given a matrix A of size $N \times N$. Your task is to output another matrix B of equal size such that B is balanced and $B[i][j] \ge A[i][j]$ for all $1 \le i, j \le N$. Furthermore, your B must have the minimum possible sum of entry values.

Input

The first line of input contains an integer N, the number of rows and columns of the matrix ($1 \le N \le 50$).

Each of the following N lines contains N integers. Together they describe the matrix A. It is guaranteed that $0 \le A[i][j] \le 35\,000$ for all $1 \le i, j \le N$.

Output

On the first line, print the sum of the values of the balanced matrix B you found. On the next N lines, print the balanced matrix in the same format as given in the input.

Any output matrix that meets the constraints described in the statement will be accepted. The values of the output matrix are not constrained in any way (specifically, they may exceed the value 35 000).

Example

standard input	standard output
4 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1	16 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Problem B. Entanglement

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	256 mebibytes

Consider an array A of length N and an array B of length M. An *entanglement* of these two arrays is a matrix C of size $N \times M$ such that for all $0 \le i \le N-1$ and $0 \le j \le M-1$, at least one of the following conditions holds: $C[i][j] = A[i]$ or $C[i][j] = B[j]$.

You are given a matrix C of size $N \times M$ and a number K. Your task is to count the number of pairs of arrays (A, B) such that:

- A has length N.
- B has length M.
- A and B consist of values from the set $\{1, 2, \ldots, K\}$.
- C is an entanglement of A and B.

Print the number of such pairs modulo $10^9 + 7$.

Input

The first line of input contains three integers N, M and K ($1 \le N, M \le 300$, $1 \le K \le N \times M$).

Each of the following N lines contains M integers separated by spaces, the j-th number on the i-th of these lines is $C[i-1][j-1]$.

Output

Print a single line containing a single integer: the number of pairs of arrays (A, B) modulo $10^9 + 7$.

Example

standard input	standard output
2 2 2 1 1 1 2	5

Problem C. Gravity

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

You are given an $N \times M$ binary matrix. Cell (i, j) contains the character "`.`" if it is free and the character "`#`" if the cell is occupied by a piece. Each maximal 4-connected component of "`#`" characters forms an indivisible piece. During the process described below, the pieces do not merge or split in any way. All cells that are part of the same piece will move in exactly the same way.

The pieces start falling towards the floor (the last row of the matrix) with equal speeds. The pieces move down without any rotations. Every second, all pieces try to move down one row. If this motion results in a piece crossing the lower boundary of the matrix, that piece stops in place instead. Similarly, if this motion results in a piece overlapping with another piece (note that this can only happen if the latter piece is not moving), then the former piece also stops in place. In other words, pieces stop falling when they hit the floor or when they hit another piece.

Output the final state of the pieces after all of them stop falling.

Input

The first line of input contains two integers N and M ($1 \leq N, M \leq 2000$).

The next N lines describe the matrix. Each of them contains M characters which are either "`.`" or "`#`". The characters denoting cells on the same line are **not** separated by any whitespace.

Output

Print the resulting matrix after all pieces have finished falling. The matrix must be printed in the same format as given in the input, except for the line containing the matrix dimensions.

Example

standard input	standard output
10 10 ....######.. ..#....#.. ..#.#..#.. ..#..#.#.. ..#....#.. ..######.. ....#....#.. ....#..	....######.. ..#....#.. ..#....#.. ..#....#.. ..#.##.#.. ..######.. ....#.. ..#....#..

Problem D. Infinite Pattern Matching

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Consider the infinite binary string I formed by concatenating the binary representations of all the strictly positive integers in increasing order: $I =$ "11011100 . . . ".

You are given a binary string A. Your task is to find the smallest integer L such that A is a suffix of $I[1 \ldots L]$.

Input

The only line of input contains the binary string A, $1 \leq |A| \leq 55$.

Output

Print a single line with a single integer: the number L.

Examples

standard input	standard output
11	2
011011	42

Problem E. Inheritance

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

You are the proud owner of K apples. You want to leave them as inheritance to your children and grandchildren. You have N children. Each of these children has children of their own. The array $grand[]$ describes this: its i-th value is equal to the number of children your i-th child has. So, your total number of grandchildren is equal to $grand[1] + grand[2] + \ldots + grand[N]$. It is guaranteed that $grand[i] \geq 1$ for all $1 \leq i \leq N$.

You must decide how many apples you will give to each of your children. You must distribute all K apples, none may remain. Also, you many only distribute whole apples. After you distribute the apples among your children, they will redistribute the apples between their children. Because you want to be as fair as possible, you want to find the minimum value $DIFF$ such that:

- The difference between the maximum number of apples given to a child and the minimum number of apples given to a child does not exceed $DIFF$.
- The difference between the maximum number of apples given to a grandchild and the minimum number of apples given to a grandchild does not exceed $DIFF$.

Because your children are fair as well, you may assume that they will redistribute the apples given to them in such a way that $DIFF$ will be kept as small as possible. It is, however, up to you to decide how many apples you will give to each direct child of yours.

Output the minimum possible value $DIFF$ and the quantities of apples given to each child to achieve this value.

Input

The first line of input contains two integers N and K ($1 \leq N \leq 1\,000\,000$, $1 \leq K \leq 10^{18}$).

The second line contains N integers, the i-th of them is $grand[i]$. The total number of grandchildren does not exceed $2 \cdot 10^9$.

Output

The first line must contain the value $DIFF$. The second line must contain N non-negative integers: the number of apples given to each child. The sum of these integers must be equal to K. Any distribution of apples that leads to the correct value of $DIFF$ will be accepted.

Example

standard input	standard output
2 100 1 2	15 43 57

Explanation

If you give 43 apples to your first child and 57 apples to your second child, they will split them in the following way. The first child will give all 43 apples to his only child. The second child will give 28 apples to one of her children and 29 apples to the other.

The maximum difference between apples received by children is $57 - 43 = 14$. The maximum difference between apples received by grandchildren is $43 - 28 = 15$.

Problem F. Movies

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Watching movies is a complicated affair. You have partitioned your available movies into two lists:

- the main list containing N movies,
- the secondary list containing K movies.

You therefore have a total of $N + K$ movies. Each movie has a distinct integer rating, a value from the set $\{1, 2, \ldots, N + K\}$.

Because you don't want to watch too many bad movies in a row or too many good movies in a row, you have devised the following algorithm for watching them:

- At each step, you take a movie from the main list and watch it. It then disappears from the main list.
- The movies that you select must always alternate between the worst available movie and the best available movie. The first selection must be the best movie. The movie with the lowest rating is considered to be the worst. The movie with the highest rating is considered to be the best. A movie is available only if it is contained in the main list.
- After watching the selected movie, you will choose another movie from the secondary list and insert it into the main list at any position. Once the secondary list is empty, you will stop watching movies. Note that this means that the main list will always have exactly N movies.

You want your main list to become sorted in ascending order according to movie rating. Because you are too lazy to do this properly, you won't do anything to the list itself. Instead, at each step of the algorithm, you will decide which movie to insert into the main list and where to insert it, so that the number of steps after which the main list becomes sorted is minimized.

Input

The first line of input contains two integers N and K ($1 \le N \le 100\,000$, $1 \le K \le 200\,000$).

The second line contains N integers: the main list of movies.

The third line contains K integers: the secondary list of movies.

It is guaranteed that the union of the main list ratings and the secondary list ratings equals to the set $\{1, 2, \ldots, N + K\}$.

Output

Print a single line containing a single integer: the minimum number of steps needed to sort the main list, or the value -1 if it is not possible.

Example

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

Explanation

During the first step, you will watch the movie with rating 5, because it is the best movie available. Then, you must replenish the main list. You do this by inserting the movie with rating 7 from the secondary list at the back of the main list. They now look like this:

$$\begin{aligned} \text{Main} &= [3\ 1\ 2\ 4\ 7] \\ \text{Secondary} &= [6\ 8\ 9\ 10] \end{aligned}$$

During the second step, it is time you watch the worst movie, so you watch the movie with rating 1. You replace it with the movie with rating 8, which you again place at the end of the list.

$$\begin{aligned} \text{Main} &= [3\ 2\ 4\ 7\ 8] \\ \text{Secondary} &= [6\ 9\ 10] \end{aligned}$$

During the third step, you watch the movie with rating 8 (best available), and add the movie with rating 9 at the end of the list.

$$\begin{aligned} \text{Main} &= [3\ 2\ 4\ 7\ 9] \\ \text{Secondary} &= [6\ 10] \end{aligned}$$

During the fourth step, you watch the movie with rating 2 (worst available) and add the movie with rating 10 at the end of the list.

$$\begin{aligned} \text{Main} &= [3\ 4\ 7\ 9\ 10] \\ \text{Secondary} &= [6] \end{aligned}$$

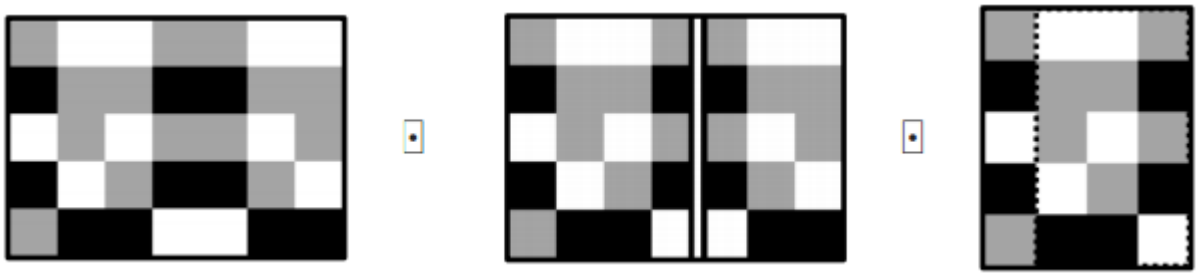
The main list is now sorted in ascending order, so the answer is 4. Note that, although all insertions in this example occurred at the end of the main list, this is not generally necessary. Movies may be inserted anywhere in the main list.

Problem G. Origami

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

You are given a piece of paper in the shape of a matrix of size $N \times M$. Every cell is colored in one of 26 possible colors which are represented by lowercase letters of the English alphabet. The piece of paper can be folded in the following way. First, you choose a vertical line between two of its columns or a horizontal line between two of its rows. This line splits the paper into two sides. You then use the line as the folding axis and fold the smaller side of the paper onto the larger one going over the axis. However, you can only do this if all the cells of the smaller side of the paper have matching colors with their corresponding cells on the other side (in other words, each cell should have the same color as its reflection over the chosen axis). If both sides of the paper are of equal size, you may fold from either side.

You notice that after any number of folding operations, you end up with a contiguous submatrix of the original piece of paper. How many different submatrices of the initial piece of paper can you obtain by doing arbitrarily many folding operations (possibly none)? Two submatrices are considered different if they occupy different coordinates in the initial matrix, even if they have identical color content.



Input

The first line of input contains two integers N and M ($N, M \geq 1$, $1 \leq N \cdot M \leq 1\,000\,000$).

Each of the next N lines contains a string of length M consisting of lowercase English letters. Together, these lines describe the given piece of paper.

Output

Print a single line with a single integer: the number of different submatrices that you can obtain by folding the paper.

Example

standard input	standard output
5 7 baabbaa cbbccbb ababbab cabccba bccaacc	2

Problem H. Qnp

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1.5 seconds
Memory limit:	256 mebibytes

You are given some digits. Your task is to find the K-th smallest integer that consists of **exactly** the digits given, modulo $10^9 + 7$. You should answer Q such queries (a query consists of digit frequencies and an integer K).

Note that integers with leading zeroes are also taken into account.

Input

The first line contains a single integer Q ($1 \le Q \le 5000$).

Each of the next Q lines contains 11 integers. The first ten denote the frequencies of digits 0, 1, ..., 9. The last one is the integer K ($1 \le K \le 10^{12}$). For each query, the total number of digits is strictly positive and does not exceed 70 000.

Output

Print Q lines. The i-th line must contain one integer: the answer for the i-th query modulo $10^9 + 7$.

Example

standard input	standard output
6 1 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 2 1 1 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 2 1 1 1 0 0 0 0 0 0 0 5 1 2 0 0 0 0 0 0 0 0 2	1 10 12 21 201 101

Problem I. Salaj

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	256 mebibytes

Given two integers V and E, we say that an array A of length E is a valid *connectivity history array* (CHA, for short) if there exists a directed graph with V vertices and E edges and a permutation P of length E such that, if we were to start with an empty graph on V vertices and add edges in the order given by permutation P, then it is true for all $1 \leq i \leq E$ that $A[i]$ is the number of strongly connected components in the graph after adding the $P[i]$-th edge.

For example, for $V = 3$ and $E = 3$:

$A = [3, 3, 3]$ is a valid CHA because there exists the graph $\{(1, 2), (1, 3), (2, 3)\}$. Regardless of the edge order chosen, the graph will always have three strongly connected components.

$A = [3, 3, 1]$ is a valid CHA because there exists the graph $\{(1, 2), (2, 3), (3, 1)\}$. If we add the edges in exactly this order, the graph will have three strongly connected components after the first two edges, and a single strongly connected component after the third edge is added.

$A = [3, 2, 1]$ is not a valid CHA because there does not exist a graph with three edges on three vertices and an order of adding edges such that the number of strongly connected components decreases after each added edge.

Note that the graph is not allowed to contain self-loops or multiple edges (but if the edge (x, y) exists, the edge (y, x) may also exist).

You are given Q queries, each consisting of a pair of integers (V, MAX). Your task is to count the number of valid CHAs for each pair (V, E) such that $1 \leq E \leq MAX$.

Input

The first line contains a single integer Q, the number of queries ($Q \leq 10$). In the next Q lines, the i-th line describes the i-th query and contains three integers V_i, MAX_i and MOD_i ($1 \leq V_i \leq 50$, $1 \leq MAX_i \leq V \cdot (V - 1)$, $1 \leq MOD_i \leq 10^9$).

Output

Output Q lines, one line per query. The i-th line must contain MAX_i integers denoting the answers for the i-th query, taken modulo MOD_i. The j-th integer on the i-th line must denote the number of valid CHAs for the pair (V_i, j).

Example

standard input	standard output
2 5 10 666013 6 9 10	1 2 4 9 21 50 110 209 351 546 1 2 4 9 1 1 6 0 7

Problem J. Taxi

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

You are given an undirected tree on N vertices. Each edge has a length, which is a strictly positive integer. M taxis and M customers will appear in this tree, each taxi and each customer appearing in exactly one node. It is possible that a node will contain multiple taxis and/or multiple customers.

A taxi-app will match customers with taxis. These days, the customer must pay for the distance that the taxi travels just to pick them up. The taxi-app is mischievously greedy, so it will match customers with taxis such that the total distance travelled by taxis to their respective customers is as high as possible. Note that each taxi gets assigned to exactly one customer and each customer is assigned to exactly one taxi.

There are N^{2M} different ways in which the taxis and customers may appear in the tree. For each of these ways, we can find the total distance travelled by taxis according to the mischievously greedy matching picked by the taxi-app. Your task is to add all these distances together and compute this sum modulo $10^9 + 7$.

Input

The first line contains two integers, N and M ($1 \leq N, M \leq 2500$).

Each of the next $N-1$ lines contains three integers: x, y and l. This means that there exists an undirected edge between nodes x and y of length l ($1 \leq l \leq 10\,000$). It is guaranteed that the given edges form a tree.

Output

Print a single line containing a single integer: the required sum modulo $10^9 + 7$.

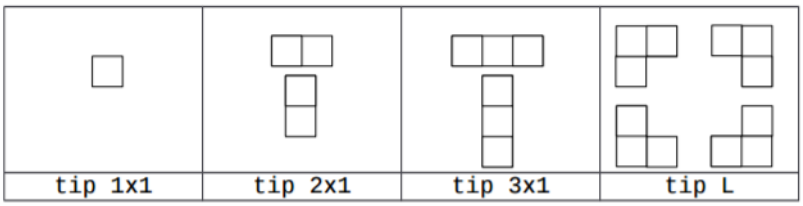
Example

standard input	standard output
5 2 4 5 9805 3 4 2001 2 3 6438 1 3 3790	10784056

Problem K. Tris

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

You are given some domino-like pieces. The following types of pieces are possible:



Note that there are only four types, and you may rotate and reflect any piece for further use. You want to place all the pieces in a matrix of size at most 800×800 so that you get a single non-self-touching cycle. Formally, this means:

- All pieces must fully fit in the matrix and be aligned with the grid.
- No two pieces may overlap.
- If a certain matrix cell is occupied by a piece, then exactly two of its four neighbours must also be occupied.
- All occupied cells are connected. In other words, you can travel from any occupied cell to any other occupied cell by only moving to adjacent occupied cells.
- The "interior" of the cycle must be a single 4-connected area.

Input

The input consists of a single line containing four integers: the number of pieces of each type (in the order they are shown in the image). It is guaranteed that each number is at least 2 and at most 100, and that at least one valid answer exists.

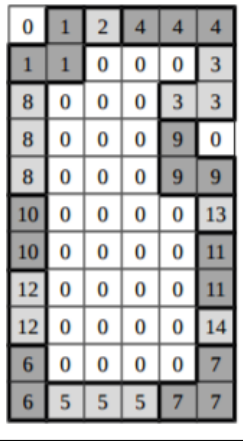
Output

The first line of output must contain two integers N and M ($N, M \leq 800$) denoting the number of rows and columns in your matrix. The next lines must describe the matrix in the following format:

- The matrix must contain integers between 0 and the total number of pieces, inclusive.
- Cells occupied by the same piece must have the same value.
- Cells occupied by different pieces must have different values.
- Cells that are not occupied by any piece must have the value 0.

If there are several valid answers, print any one of them.

Example

standard input	standard output	picture
3 4 3 4	11 6 0 1 2 4 4 4 1 1 0 0 0 3 8 0 0 0 3 3 8 0 0 0 9 0 8 0 0 0 9 9 10 0 0 0 0 13 10 0 0 0 0 11 12 0 0 0 0 11 12 0 0 0 0 14 6 0 0 0 0 7 6 5 5 5 7 7	

Problem L. Xormites

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Consider the following two-player game on a sequence of N integers. The two players move in turns. In each move, the current player selects a value which is either at the beginning of the sequence or at the end of the sequence, adds it to this player's sum and removes the value from the sequence.

This is a well-known game. However, in this task, we will deal with the case in which the players add the values to a XOR sum, not a regular sum. Initially, the XOR sums of both players are equal to 0, and when adding a new value, bitwise XOR operation is used in place of addition.

The game ends when the sequence is empty, at which point the player with the highest XOR sum wins. Note that it is also possible for the game to end in a draw. Figure out the outcome of the game, considering the players behave optimally.

Input

The first line contains an integer T, the number of test cases ($1 \le T \le 12$). Each test starts with a line containing an integer N ($1 \le N \le 50\,000$), followed by another line containing a sequence of N positive integers ($1 \le X \le 10^9$ for all integers X in the sequence).

Output

Print T lines, one per test case. The i-th line must contain the answer for the i-th test. The answer must be one of the following:

- "`First`" if the first player to move wins.
- "`Second`" if the second player to move wins.
- "`Draw`" if the game ends in a draw.

Example

standard input	standard output
3 2 3 3 2 3 5 3 4 4 4	Draw First Second