Problem A. Arcs

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

There are N points on a circle, represented by numbers a_i, where a_i — direction from center to point i, measured in 1/100th parts of degree.

Your program must choose a set of pairs of points so that:

- Each pair of points defines an arc of a circle, shorter of two possible arcs.
- For a pair of points lying on the diameter, it is allowed to choose any of the arcs.
- Each point belongs to at most one pair.
- Arcs defined by different pairs do not have common points.
- The total length of all arcs, measured in 1/100th parts of degree, is as large as possible.

Input

First line of the input contains one integer N $(2 \le N \le 5000)$. Second line contains N integers a_i $(0 \le a_i < 36\,000)$ — directions from center to points, measured in 1/100th parts of degree.

Output

Output must contain two integers S and M, where S – largest total arc length, M – number of selected pairs. Next, output must contain M pairs of integers – pairs of point *indices* corresponding to the ends of each arc. Indices start with 1. The order of points in pair may be arbitrary.

If there are several optimal solutions, output any of them.

Examples

standard input	standard output
4 1 2 18000 18001	35998 2 2 3 4 1
6 100 200 12000 12100 24000 24100	35700 3 2 3 4 5 6 1

Problem B. Bytica's Algebra

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

Bytica is preparing for PhD exams in algebra. She studies next problem:

Given $N + 1$ integers $a_0 \ldots a_N$ and prime P.

Count, how much non-negative integers z, strictly less than P, have a next property: the value of the polynomial $a_n \cdot z^n + a_{n-1} \cdot Z^{N-1} + \ldots + a_2 \cdot z^2 + a_1 \cdot z + a_0$ is divisible by P?

Input

The input consists of several test cases.

First line of the each test case contains two integers N and P ($0 \leq N \leq 100$, $2 \leq P \leq 10^9$, P is prime). Second line contains $N + 1$ integers; i-th of those integers is a_{i-1} ($-10^9 \leq a_i \leq 10^9$).

The input is terminated by case with $P = 0$, which should not be processed. You may assume that sum of all N in the the input does not exceed 1500.

Output

For each test case print the answer to the problem.

Example

standard input	standard output

2 3
1 2 1
2 3
1 2 6
0 0
	1
	1

Problem C. Cutting Brackets

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

For a string of parentheses '`(`' and '`)`', let us define *cutting* as follows. Pick a random pair of characters '`(`' and '`)`' such that '`(`' is to the left of '`)`' and delete them. Selection probability is distributed uniformly across all possible pairs. Repeat cutting until no suitable pair left.

Given a string of parentheses, your program must calculate the probability of the above process to finish with an empty string.

Input

Input contains a string of parentheses S $(1 \le |S| \le 36)$.

Output

Output must contain a single floating point number – probability with absolute error of no more than 10^{-7}.

Examples

standard input	standard output
`())`	`0`
`(())`	`1.0`
`()()()`	`0.333333333`

Problem D. Drawing Clusters

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

There are n circles on the plane. Bytica have k colors and paints the plane according to the following rules. Consider *cluster* as the finite part of plane with nonzero area, surrounded by a set of arcs

- Each cluster is painted with a single color or not painted at all.
- Two clusters sharing a part of the boundary with non-zero length cannot be painted by same color (but they both can be unpainted).

Find the maximum number of clusters that can be painted.

Input

First line of the input contain two integers n and k ($1 \le n \le 20$, $1 \le k \le 10^9$) — number of circles and number of colors, respectively. Then n lines follow, each containing three integers x, y and r ($-1000 \le x, y \le 1000$, $1 \le r \le 1000$) — coordinates of center of circle and its radius, respectively.

Output

Print one integer — maximum number of clusters that can be painted.

Example

standard input	standard output

2 1
1 0 1
2 0 1
	2

Problem E. Eyesight Development

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

Emma writes a 2D game engine. One of the tasks is to determine parts of the scene visible by the gamer from a given position.

Let us consider a set of rectangles with sides parallel to the coordinate axes: $P_i = \{(x, y) : a_i \le x \le b_i, c_i \le y \le d_i\}$.

Rectangle P_i is *visible* from the point A, if there exists some point $B \in P_i$ such that eyesight segment AB does not contain points belonging to other rectangles.

Your program must determine *indices* of rectangles that are visible from the point $(0, 0)$.

Input

Input contains integer n followed by $4 \times n$ integers: a_i, b_i, c_i and d_i ($2 \le n \le 10^5$, $-10^6 \le a_i \le b_i \le 10^6$, $-10^6 \le c_i \le d_i \le 10^6$).

Rectangles do not intersect each other and do not contain point $(0, 0)$.

Output

Output must contain indices of visible rectangles in ascending order. Indices start from 0.

Examples

standard input	standard output

5
-3 3 -2 -1
2 8 4 6
1 4 2 3
-4 -1 2 8
1 7 -5 -3
	0
	2
	3

5
0 4 3 7
-1 2 1 2
1 5 -3 -2
0 0 -6 -5
0 0 -4 -2
	1
	2
	4

Problem F. Ferry Express

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

"Ferry Express" is a simple arcade game. The main scene of the game is a river that is represented as a 1-dimensional line. Coins fall down from height H to this line. You must catch them in his boat when they touch the river line.

The boat is represented as a *closed* segment (containing its boundary points) of length L moving along line from the position with point 0 at left end of ferry.

Boat can move only from left to right with a fixed integer speed. It is known that i-th coin falls down to point P_i with a speed V_i.

A coin is caught if by the moment it crosses the river line, this point belongs to the boat.

Your program must, given H, L, P_i, and V_i, determine the minimum possible *integer* speed of the boat maximizing the number of caught coins.

Input

Input contains integers H, L, and n, followed by n pairs of integers P_i, V_i $(0 < (H, L, V_i) \leq 10^6$, $0 \leq P_i \leq 10^6$, $1 \leq n \leq 2 \cdot 10^5)$.

Output

Output must contain a single integer – the speed of the boat.

Examples

standard input	standard output

10 2 4
2 1
3 8
4 7
5 8
	2

10 2 4
9 3
8 2
6 3
8 2
	0

Problem G. Game Frotnite

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

Vasya created a two-dimensional game Frotnite in Battle Royale genre. The game is played on a square field of N by N cells. Each cell is either empty (represented by '`.`') or occupied by wall (represented by '`#`').

The player is located in one of the empty cells. Every second the player can stay in place or move to an adjacent empty cell up, down, left or right. After player moves, all cells along the perimeter of the game field are removed (so field size is reduced by 2 along each axis). If the player was located on one of the removed cells, he dies.

Your program must, for each empty cell of the field, calculate maximum number of seconds the player can survive if he starts the game from that cell and plays optimally.

Input

The first line of input contains a single integer N ($1 \le N \le 500$). The next N lines contain one string of N characters each – representation of the game field.

Output

The output should contain N lines of N numbers, where the j-th number in the i-th line indicates the maximum survival time of a player when starting from a cell with coordinates (i, j). If the corresponding cell is not empty, output zero.

Examples

standard input	standard output

5
.....
.....
.....
.....
.....
	1 2 3 2 1
	2 3 3 3 2
	3 3 3 3 3
	2 3 3 3 2
	1 2 3 2 1

5
.....
.#.#.
.....
.#.#.
.....
	1 1 3 1 1
	1 0 3 0 1
	3 3 3 3 3
	1 0 3 0 1
	1 1 3 1 1

Problem H. Height Growing

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

Given a Treap with real number used as key.

Reminder that Treap is a balanced binary search tree using random numbers. In addition to the key, each node has a value called priority. Here, the key and priority are real numbers from 0 to 1. In Treap, the following conditions are always maintained.

- For each node, nodes below the left child have smaller keys than themselves
- For each node, the nodes below the right child have larger keys
- For each node, the child node has a lower priority than itself

The insertion operation is performed as follows. Let the key to be inserted is x.

The priority p of the new node to be inserted is randomly chosen from the uniform distribution of 0 to 1 . Similar to a normal binary search tree, first the priority is ignored and a new node is inserted. Rotation operations such as lifting a new node up are repeated as many times as necessary to satisfy the priority condition.

Suppose N keys are randomly selected from the uniform distribution of 0 to 1 and inserted into an empty Treap. Find the probability that the height will eventually be h for each $h = 0, 1, \ldots N - 1$. Note that real numbers are treated with sufficient accuracy, and the probability that the priorities of two nodes are equal is 0, for example.

The height of treap is defined as maximum number of edges that must be passed from the root node to each node.

Input

Input consists of one integer N ($1 \le N \le 30\,000$).

Output

Print N lines. In i-th of them print the probability that the height becomes $i - 1$ with absolute error 10^{-5} or better.

Example

standard input	standard output
3	0.00000000 0.33333333 0.66666667

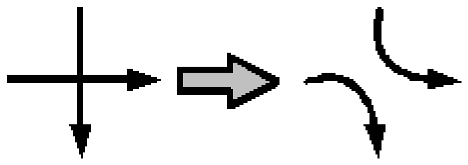
Problem I. Improving Hackersburg

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

Hackersburg is surrounded by Hackersburg Ring Road (HRR) which separates business and cultural center from the rest of the city. The road itself is nothing more than a traffic pattern, which all vehicles should obey.

Now the driverless cars are introduced in Hackersburg. But that design have one shortcoming — first cars were only able to turn to the right. Such a bizarre engineering solution had a serious impact on the design of HRR: it is possible to depart from any point on the road, make one or more circles around the business center and return to the starting point place, having visited all the road and making right turns only.

The traffic is one-way everywhere, so driving around Hackersburg is simple. The only complication is a necessity to pass intersections. When passing an intersection, vehicles must move straight ahead (i.e. turns on intersections are forbidden).



Recent upgrade fixed the bug, so cars are now capable of turning either right or left. After providing all citizens with new cars, it became possible to convert some intersections into turn pairs (see picture) to further simplify the life in Hackersburg. THe conversion must preserve the direction of traffic on each road segment.

The mayor of Hackersburg calls for your help. You must plan how to convert a maximum possible number of intersections into turn pairs, without breaking BRR connectedness.

Input

First line of the input file contains integers WH $(2 \leq W,\ H \leq 20)$. Following H lines contain W symbols each and represent a map of HRR

Used symbols are: '│' (ASCII 179), '─' (ASCII 196), '┌' (ASCII 218), '└' (ASCII 192), '┘' (ASCII 217), '┐' (ASCII 191), '┼' (ASCII 197, represents intersection), ' ' (ASCII 32).

NOTE: the input files are in DOS (aka CPP866) encoding. You can download actual input files for the samples at address **http://opentrains.mipt.ru/ ejudge/mw2020fday4samplesJ.zip**.

It is guaranteed that somewhere on the map there exists an area of business and cultural center, which is always seen by the right hand when moving along the BRR.

Output

First line of output file should contain an integer K — maximal number of crossings that can be converted into turn pairs.

Following K lines should contain a sequence of integer pairs, separated by spaces — coordinates of these crossings. Each pair consist of row and column numbers of input matrix respectively. Numeration starts from 0, so upper-left corner of input matrix has coordinates 00.

Examples

standard input	standard output
4 12	0

5 9	2
	2 6
	2 7

Problem J. Just Interesting

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

Consider a permutation $p_1 \dots p_N$ *interesting*, if for given s t $p_s = t$, and for K given pairs of a_i and b_i $p_{a_i} < p_{b_i}$. Given all those parameters, calculate number of the interesting permutations.

Input

First line of the input contains four integers N, K, s and t ($1 \le N \le 2000$, $0 \le K \le N$, $1 \le s, t \le N$. Each of the following k lines contains a_i and b_i ($1 \le a_i, b_i \le N$, $a_i \ne b_i$, all a_i are distinct).

Output

Print one integer — number of interesting permutations modulo $10^9 + 7$.

Examples

standard input	standard output
3 1 3 3 1 3	2

5 2 2 2
1 3
3 1
	0

Problem K. Kirchhoff

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

You are given a resistor circuit. Your task is to find the resistance between the given pairs of nodes.

The resistor circuit contains N nodes and a set of resistors. A resistor with a resistance of R_{ij} may be soldered in between every two i and j nodes. When the ohmmeter is connected to two nodes on the circuit, the resistance those nodes is measured. The ohmmeter creates a potential difference between these nodes and measures that difference and the current passing through the circuit. Thus, according to Ohms law, the circuit resistance is found to be equal to

$$R_c = \frac{f_t - f_s}{I_c}$$

Where t and s are the circuit nodes to which the ohmmeter is connected, , f_t is the electric potential in the node i, and I_c is the current passing through the circuit. Also the first law of Kirchhoff sounds as:

Taking $I_{ij} + I_{ji} = 0$, for all $i \neq s, t$, $\sum_{j=1}^{N} I_{ij} = 0$ holds true.

You are given a resistor circuit. You must find the resistance for the given pairs of nodes.

Input

The first line of the input file contains two integers N and M, number of nodes in the circuit and number of resistors, respectively ($2 \leq N \leq 300$, $M \geq 1$). Each of following M lines contains a desctiption of one resistor: three integers A_i, B_i and C_i, where A_i and B_i are indices of nodes connected by a resistor and C_i is the value or resistor's resistance ($1 \leq A_i, B_i \leq N$, $1\ eC_i \leq 1000$).

The next line contains an integer Q, which is the number of queries ($1 \leq Q \leq 4.5 \cdot 10^4$). Then Q lines follow, describing the queries. Each query consists of two different integers S_j and T_j — the indices of nodes the resistance between which is to be measured ($1 \leq S_j, T_j \leq N$). There is no more than one resistor connecting any two nodes, and no resistor connects a node to itself. You may assunme that the circuit is connected, i.e. every node is linked to all other nodes via a sequence of resistors.

Output

Print Q lines, each line containing the measured value of resistance between nodes S_k and T_k with absolute error 10^{-6} or better.

Examples

standard input	standard output

3 3
1 2 1
2 3 2
1 3 1
3
1 2
2 3
3 1
	0.75
	1.0
	0.75