Problem A. Wheel

Input file:	bar.in
Output file:	bar.out
Time limit:	3 seconds
Memory limit:	256 mebibytes

One popular game uses a special wheel as game equipment. The upper surface of the wheel is a circle divided into N equal sectors. Each sector is colored in either red or blue. Wheel coloring is considered correct if for each M consecutive sectors of the wheel, there are no more than R blue sectors among them. Two wheels are considered different if they cannot be rotated so that their colorings become equal.

Given N, M and R, find the number of different correct colorings of the wheel modulo $1\,000\,000\,007$.

Input

The first line of input contains three integers N, M and R ($1 \le R \le M \le 6$, $2 \le N \le 10^9$, $M \le N$).

Output

The first line of output should contain one integer — the number of different correct colorings modulo $1\,000\,000\,007$.

Examples

bar.in	bar.out
2 2 1	2
7 5 3	11

Problem B. Garden

Input file:	garden.in
Output file:	garden.out
Time limit:	3 seconds
Memory limit:	256 mebibytes

Petya lives in a house located at point $(0, 0)$ on the Cartesian plane. There is a garden around the house: there's a peach tree at every point with integer coordinates not exceeding N by absolute value except $(0, 0)$.

Recently, peaches ripen on all the peach trees. Now, Petya wants to gather them all. He starts at his house (point $(0, 0)$) looking in the direction of the tree $(1, 0)$. Then, he starts counting the trees he sees in counter-clockwise direction up to the K-th tree. After that, Petya memorizes the position of that tree and moves along the ray towards it, gathering peaches from all trees on his way. Finally, he returns to his house at $(0, 0)$ and repeats the steps above, starting the count from the first tree after the memorized one in counter-clockwise direction. This procedure repeats until Petya gathers peaches from all the trees.

Note that when counting, Petya does not see any trees which are directly behind some other trees when looking from $(0, 0)$. Petya also does not count trees from which he has already gathered peaches.

Given N and K, find the last tree from which Petya will gather peaches.

Input

The first line of input contains two integers N and K ($1 \le N \le 10^5$, $1 \le K \le 10^5$).

Output

The first line of output should contain two integers which are x- and y-coordinates of the last tree from which Petya will gather peaches.

Examples

garden.in	garden.out
1 1	1 -1
2 2	2 0
10 4	-8 -6
12345 67890	11299 1509

Problem C. Peaceful Knights

Input file:	knights.in
Output file:	knights.out
Time limit:	3 seconds
Memory limit:	256 mebibytes

Thirty-two knights are placed on the standard 8×8 chessboard. A move consists in moving a knight by chess rules: two cells along any horizontal or vertical direction and one cell along the orthogonal direction. Two knights can not be located on the same cell at any moment.

Given the initial positions of the knights, achieve a *peaceful* placement in minimal number of moves. Knights' placement is considered peaceful if no knight attacks another knight, i. e. no knight has a move to the position of another knight.

Input

Input consists of one or more test cases (no more than 100).

Each test case consists of eight lines with eight symbols in each line. These lines describe the initial positions of the knights; a free cell is denoted by '.', while a cell with a knight is denoted by 'N'. Consecutive test cases are separated by an empty line.

Output

The answer for each test case should start with a line containing K — the minimal number of moves. Next K lines should contain the sequence of moves which lead to a peaceful placement. A single move is denoted by the starting cell and the destination cell, separated by a dash. Rows are numbered '1' through '8' in the order from bottom to top, that is, the first line of each test case has number '8', the second line has number '7', and so on. Columns are numbered 'a' through 'h' in the natural left-to right order.

If there are several solutions for a certain test case, you can output any of them. Consecutive test cases should be separated by an empty line. Adhere to the sample output below as close as possible.

Examples

knights.in	knights.out
.NN..N.N N.N.N.N. ..NN.N.N N.N.N.N. .N.N.N.N N.N.N.N. .N.N.N.N N.N.N.N. N.N.N.N. ..NN.N.N N.N.N.N. .N.N.N.N N.N.N.N. .N.N.N.N N.N.N.N. .NN..N.N	2 c6-d8 c8-b6 6 d3-f2 f2-d1 c1-d3 e6-d8 d8-b7 c7-e6

Problem D. Logs

Input file:	log.in
Output file:	log.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

In a far a way land on the Cartesian plane, there's a lake in the form of a rectangle with vertices in $(0, 0)$, $(W, 0)$, (W, H) and $(0, H)$. There is an island in the lake which has the shape of a strictly convex polygon. Vasya has two logs of length L. He wants to build a bridge that will be stable and will connect the mainland (the outer region of the rectangle) with the island (the inner region of the polygon). In a stable bridge, the first log has its endpoints on the land, and the second log has one endpoint on the first log and another endpoint on the land.

Given coordinates of the lake and the island, find the minimal length of logs L such that Vasya can construct a stable bridge between the mainland and the island. The thickness of logs is negligible.

Input

The first line of input contains three integers N, W and H ($3 \le N \le 20$, $3 \le W, H \le 1000$).

Each of the next N lines contains two integers x_i and y_i — coordinates of i-th vertex of the polygon corresponding to the island ($0 < x_i < W$, $0 < y_i < H$). Vertices are given in either clockwise or counter-clockwise order.

Output

In the first line of output, write a single real number — the minimal length of a log. Your answer should have no less than 7 correct digits after decimal point.

Examples

log.in	log.out
4 10 10 3 3 3 7 7 7 7 3	2.828427125
4 10 10 4 3 3 6 6 7 7 4	3.000000000

Problem E. Polyline

Input file:	polyline.in
Output file:	polyline.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

Given N^2 points located inside a unit square, connect them all with a polyline which has length no greater than $2N$.

Input

The first line of input contains an integer N ($2 \le N \le 311$). The next N^2 lines contain coordinates of points (x_i, y_i) ($0 \le x_i \le 1, 0 \le y_i \le 1$). Coordinates are real numbers given with no more than nine digits after decimal point. All given points are different.

Output

The first line of output should contain N^2 integers — the numbers of points in the order of traversing the polyline. The points are numbered 1 through N^2 in the order in which they are given in the input. If there are several answers, output any of them.

Examples

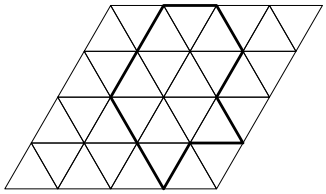
polyline.in	polyline.out
2 0 0 1 1 0 1 1 0	1 2 3 4
3 0.1 0.1 0.5 0.5 0.9 0.9 0.3 0.0 1.0 0.1 0.0 0.9 0.1 0.6 0.4 0.89 1.0 0.47	1 4 5 9 3 8 6 7 2

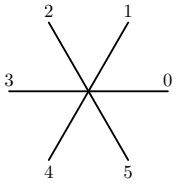
Problem F. Unfolding

Input file:	raz.in
Output file:	raz.out
Time limit:	5 seconds
Memory limit:	256 mebibytes

A piece of paper is covered by triangular grid. There is also a polygon A with vertices in the vertices of the grid and sides going along the lines of the grid. Let a *proper unfolding* be a polygon B such that

- it is cut out of polygon A along the lines of the grid,
- it is connected in the sense that one can reach any of its triangles from any other of its triangles by a sequence of moves, while a move is allowed between two triangles if they share a common edge, and
- it is an unfolding of a regular polyhedron with a side equal to the side of the grid triangle, that is, one can make a regular polyhedron of the polygon B folding paper along the grid lines.





The left picture contains the polygon from the second sample.

Given the polygon A, find the number of ways to cut a proper unfolding from it.

Input

The input contains description of the polygon A.

The first line of input contains a single integer N ($3 \le N \le 14$) — the number of segments of unit length which comprise the border of the polygon.

The next line contains N numbers: the descriptions of the segments in either clockwise or counter-clockwise order. Each segment is given by its direction — an integer from 0 to 5, as shown on the right picture. It is guaranteed that A does not contain any self-touchings or self-intersections.

Output

The first line of output should contain a single integer — the required number of ways to cut a proper unfolding.

Examples

raz.in	raz.out
6 1 1 5 5 3 3	1
10 1 1 0 5 4 5 3 4 2 2	19
3 2 4 0	0
12 5 5 5 1 1 1 1 3 3 3 3 5	25

Problem G. Road Works

Input file:	`rw.in`
Output file:	`rw.out`
Time limit:	3 seconds
Memory limit:	256 mebibytes

In a far land of Roadland, there are N cities connected by M bidirectional roads. The road network is organized in such a way that

- no road connects a city to itself,
- no two cities are connected by more than one road,
- each city can be reached from each other city by roads, and
- if cities A and B are connected by road, there exists at most one simple path from A to B which does not contain that road.

Recall that a *simple* path visits each vertex not more than once.

Each road is described by which cities it connects, as well as what number of seconds it takes initially to drive along that road.

Starting from day 1 and until day K, a single event happens every day. There are two types of events. The first type is a query of the form "what is the minimal time to get from city x to city y". The second type is an information message of the form "the road number x was repaired this day, and it now takes y seconds to drive along that road". All roads suffer damage over time, so at the beginning of each day, before any event takes place, the time to drive along each road is increased by one second.

Given the road network and the sequence of events, find the answers of the queries.

Input

The first line of input contains two integers N and M ($2 \le N \le 10^5$).

Next M lines contain three integers each: a_i, b_i and t_i. Here, a_i and b_i are the numbers of cities that the respective road connects, and t_i is the time it takes to drive along that road ($1 \le a_i, b_i \le N$, $1 \le t_i \le 10^5$). Roads are numbered 1 through M in the order in which they are given.

The next line of input contains a single integer K ($1 \le K \le 10^5$) — the number of events.

Next K lines contain the descriptions of events, one line per event. A query is given as `P` x y ($1 \le x, y \le N$), and an information message is given as `R` x y ($1 \le x \le M$, $1 \le y \le 10^5$).

All numbers in the input are integers.

Output

For each query, output a single line with a single integer on it — the minimal time it takes to drive between the cities given in the query.

Example

rw.in	rw.out
4 4 1 2 1 2 3 1 3 4 1 4 2 1 10 P 1 2 P 2 3 P 1 4 P 1 3 R 1 5 R 2 1000 P 1 2 P 2 4 R 2 1 P 1 4	2 3 8 10 7 9 21

Problem H. Checkers

Input file:	sha.in
Output file:	sha.out
Time limit:	3 seconds
Memory limit:	256 mebibytes

Vasya and Petya enjoy playing checkers very much. In the current game, the white player has three kings left on the board, and the black player has just one king left.

The board of the game is a standard 8×8 chessboard; cell a1 is black.

There are two ways to move a king.

- A *simple move* consists in moving a king by some positive number of cells along any diagonal provided that the destination cell, as well as all cells in between, are empty.
- A *jump* consists in making a king jump over an opponent's piece on the board and land somewhere beyond it on an empty cell. The two pieces involved and the destination cell must be on the same diagonal, and there should be no other pieces between the king's cell and the destination cell except the opponent's piece which is jumped over. Jumping is mandatory: one can not make a simple move if a jumping move is available.

  Multiple-jump moves are possible if, when the jumping piece lands, there is another piece which it can jump over. Again, if some destination cell gives an opportunity for another jump, the destination cell should be chosen only among such cells. This restriction, however, is *not* applied recursively: for example, if a certain first jump leads to a 3-jump move, and another possible first jump leads to a 2-jump move, any of these first jumps can be chosen.

  After a sequence of jumps, all opponent's pieces which were jumped over are removed from the board simultaneously. It is not allowed to jump over any opponent's piece more than once during a multiple-jump move.

The white player moves first. A player loses if he has no valid move. If each player makes 32 moves and the game did not end, it is considered a draw.

Given the initial positions of the kings, find the minimal number of turns required for white player to win, assuming that both players play optimally.

Input

The input contains one or more test cases. The first line of input contains a single integer — the number of test cases. No two test cases give the same starting positions.

Each test case consists of two lines. The first of these lines contains the positions of white kings. The second line contains the position of the black king. A king's position is given by two characters — the number of column ('a' through 'h') and the number of row ('1' through '8'). It is guaranteed that all kings start on black cells.

Output

For each test case, output a single line containing the minimal number of white player's turns that will result in their victory, assuming that both players play optimally. If white player can not win, output -1 instead. If the white player actually wins, the black player's goal is to make the game as long as possible.

Example

sha.in	sha.out
4 e3 f2 b8 g1 a7 c5 a3 a1 a7 b8 d8 a1 a1 c1 d2 h6	1 2 -1 2

Problem I. Teleporters

Input file:	`teleports.in`
Output file:	`teleports.out`
Time limit:	4 seconds
Memory limit:	256 mebibytes

In the star system of Epsilon Eridani, there are N planets. There's a teleporter on each of them; teleporters allow fast traveling between planets. However, not all pairs of teleporters are connected.

The teleporter network is organised as follows. First, a positive integer X is chosen. After that, for every planet $i = 1, 2, \ldots, N$, the teleporter on planet i gets the number $X + i - 1$. Now, teleporter a is connected with teleporter b if the greatest common divisor of a and b is greater than one.

Given the number of planets N, find such X that the teleporter network is connected, that is, one can get from any planet to any other traveling by pairs of connected teleporters.

Input

The first line of input contains a single integer N ($1 \le N \le 10^5$).

Output

The first line of output should contain the integer X without spaces and leading zeroes. This number should be 1 to 100 000 digits long. In case of multiple answers, any of them will do. If there's no such X that the network is connected, write "`No solution`" instead. It is guaranteed that if the answer exists, there's an answer no more than 100 000 digits long.

Examples

teleports.in	teleports.out
2	No solution
100	1306976361738009591601284672880050414

Problem J. Triangle

Input file:	`triangle.in`
Output file:	`triangle.out`
Time limit:	13 seconds
Memory limit:	256 mebibytes

N points are located on the plane. Three different points are chosen randomly; all sets of three points have equal probability of being chosen. These points are then connected by line segments, and the perimeter of the resulting triangle is calculated. Note that the triangle may be degenerate.

Given the coordinates of points, find the expectation of the perimeter of the resulting triangle. The *expectation* of the perimeter is the sum $\sum_{\triangle} p_{\triangle} \cdot f_{\triangle}$ where $p_{\triangle}$ is the probability that the triangle $\triangle$ is chosen and $f_{\triangle}$ is its perimeter.

Input

The first line of input contains two integers H and W ($1 \le H, W \le 700$). Next H lines contain W symbols each; j-th symbol of i-th line is '`1`' if there is a point with coordinates (i, j), and '`0`' otherwise. There are at least three points present.

Output

The first line of output should contain one real number — the expectation of triangle perimeter. Your answer should be accurate to at least six digits after the decimal point.

Examples

triangle.in	triangle.out
11 20 10000000001000000000 00000000000000000000 00000000000000000000 00000000000000000000 00000000000000000000 00000000000000000000 00000000000000000000 00000000000000000000 00000000000000000000 00000000000000000000 10000000000000000000	34.142135624
3 3 101 010 101	5.794112550