Problem A. Attractive Flowers

Time limit: 1 second
Memory limit: 512 megabytes

Maxim wants to give a bouquet of flowers to his girlfriend Ira for her birthday.

The flower shop near his house sells flowers of n types. Maxim has found out that the store has a_i flowers of the i-th type. He knows that Ira likes odd numbers. Therefore, Maxim has decided that there should be an odd number of flowers of each type in the bouquet, and the total number of flowers in the bouquet should also be odd.

Help Maxim determine the maximum number of flowers the bouquet can consist of.

Input

The first line contains an integer n — the number of types of flowers that are sold in the store ($1 \le n \le 100\,000$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ — the number of flowers of each type ($1 \le a_i \le 1000$).

Output

Print one number — the maximum number of flowers the bouquet can consist of.

Examples

standard input	standard output
3 3 5 8	15
3 1 1 1	3

Problem B. Blocking the View

Time limit: 2 seconds
Memory limit: 512 megabytes

Anton is developing a 2D graphics engine for video games. It must hit the market. In order to display the objects correctly, he needs to understand whether one object blocks the view to another one if the player is looking in some particular direction.

For the prototype Anton considers objects to be non-intersecting segments on the plane. The segment a blocks the view to the segment b in direction of the vector $\vec{v}$, if there are such points A on a and B on b, that the vectors $\overrightarrow{AB}$ and $\vec{v}$ are co-directed. In other words, there is a point on a such that, if it starts moving along the direction of $\vec{v}$, it would hit the segment b.

Anton is busy looking for investors to his projects, so he asks you to implement this part of the graphics engine.

Input

The first line of input contains an integer n, the number of tests that follow ($1 \le n \le 50\,000$).

Each of the following n lines contains 10 integers: ax_1, ay_1, ax_2, ay_2, bx_1, by_1, bx_2, by_2, vx, vy — coordinates of the endpoints of the first segment, coordinates of the endpoints of the second segment, and coordinates of the view direction vector. All coordinates do not exceed 10^6 by their absolute values. It is guaranteed, that each segment has positive length, and that the direction vector $\vec{v}$ is non-zero.

Output

For every test print "`Yes`", if the first segment blocks the view to the second segment in the given direction, if it doesn't, print "`No`".

Example

standard input	standard output
2 0 2 1 1 2 2 3 1 1 1 0 2 1 1 2 2 3 1 -1 -1	Yes No

Problem C. Fermat's Last Theorem

Time limit: 4 seconds
Memory limit: 512 megabytes

As you probably know, for all positive integers a, b, c and n with $n \geq 3$ the following inequality holds: $a^n + b^n \neq c^n$. But all existing proofs of this fact are hard to verify, so a group of software engineers has decided to write their own proof that will be, in their opinion, easier to verify.

This group has written a program that iterates over all quadruples of positive integers (a, b, c, n) such that $n \geq 3$, in increasing order of the maximums of its elements, and in case of equality of maximums — in the lexicographical order.

Thus first the quadruple $(1, 1, 1, 3)$ will be considered, then the quadruple $(1, 1, 2, 3)$, and so on. And, for example, the quadruple $(3, 3, 3, 3)$ will be followed by the quadruple $(1, 1, 1, 4)$.

For each quadruple the program compares the values $a^n + b^n$ and c^n, and prints the corresponding inequality: $a^n + b^n > c^n$ or $a^n + b^n < c^n$.

Now the software engineers want to verify their proof. They ask you to repeat their calculations and output the inequalities printed by their program, from the l-th to the r-th one, inclusive.

Input

The first line contains two integers l and r ($1 \leq l \leq r \leq 10^{12}$; $r - l \leq 10^4$).

Output

Output the part of the printed proof, from the l-th inequality to the r-th one, each on a separate line. To denote exponentiation, use a caret ('^', the ASCII character with code 94). Don't output any spaces.

Example

standard input	standard output
1 4	1^3+1^3>1^3 1^3+1^3<2^3 1^3+1^3<3^3 1^3+2^3>1^3

Problem D. Guess the Path

Time limit:	1 second
Memory limit:	512 megabytes

This is interactive problem.

You are going to take part in robot programming contest. You will have to launch robots to a special field that has detectors at some cells. The jury has chosen one of the shortest paths on the field and you have to find it.

The field is a grid rectangle with m rows and n columns. Let us denote the cell at the i-th row and the j-th column as (i, j) ($1 \le i \le m$, $1 \le j \le n$). Every path that the robot can be launched along must start at the top-left cell of the grid which has coordinates $(1, 1)$, pass some cells of the field, and end at the bottom-right cell which has coordinates (m, n).

In one step the robot can move one cell down, or one cell to the right. Therefore if the robot is at the cell (i, j), it can move to the cell $(i + 1, j)$ or to the cell $(i, j + 1)$. The robot cannot move outside of the field. The path of the robot consists of $m + n - 2$ steps, so it moves from the cell $(1, 1)$ to the cell (m, n) visiting some other cells.

The jury has chosen one of the possible robot paths. This path is not known to participants. Each cell that belongs to the path chosen by the jury has special detector installed in it. When the robot enters the cell with the detector, the detector reports that the cell was visited.

The participant can launch the robot along any allowed path from $(1, 1)$ to (m, n). After every launch the jury tells the participant which cells that have detectors installed have reported that the cell was visited. Your task is to launch the robot at most 10 times, and find out the path that the judges have chosen.

Interaction Protocol

First your program receives two integers m and n — the size of the field for the robot ($1 \le m, n \le 1000$, $m + n > 2$).

After that your program must make queries, each query represents one launch of the robot.

To make a query output "`?` s", where s is the string that denotes the path that the robot must follow. The string must have a length $n + m - 2$ and consist of '`D`' and '`R`' characters. If the i-th character is '`D`' on the i-th step the robot moves down, if the i-th character is '`R`', it moves to the right. The robot starts its path in a cell $(1, 1)$, and after all steps it must be in the cell (m, n).

As the answer the jury outputs an integer t — the number of cells with detectors that have reported the robot has passed them ($2 \le t \le m + n - 1$). The following t lines contain two integers r_i and c_i each — the coordinates of the cells (r_i, c_i) with detectors that have reported the robot has passed them (for all $1 \le i \le t$ the inequalities $1 \le r_i \le m$ and $1 \le c_i \le n$ hold). It is guaranteed that all t cells are distinct. They are printed by increasing of r_i, and for equal r_i by increasing of c_i.

After you detect the chosen path, your program must print the line "`!` s" where s is the string that specifies the chosen path in the format described above.

If your program makes more than 10 queries, makes incorrect query, or answers the path incorrectly, it would get the verdict "`Wrong Answer`".

Example

standard input	standard output
3 4 3 1 1 3 3 3 4 4 1 1 2 2 2 3 3 4	 ? DDRRR ? DRRRD ! RDRDR

Note

Print new line character after each line. After each action your program must flush standard output.

If you use "`writeln`" in Pascal, "`cout << ... << endl`" in C++, "`System.out.println`" in Java, "`print`" in Python, "`Console.WriteLine`" in C#, standard output is automatically flushed, no additional action is required. If you are using other way to output data, you should flush standard output. Note that even if you flush output, you absolutely must print new line character. To flush output you can use "`fflush(stdout)`" in C and C++, "`flush(output)`" in Pascal, "`System.out.flush()`" in Java, "`sys.stdout.flush()`" in Python, "`Console.Out.Flush()`" in C#.

Explanation

The picture 1 shows the path chosen by the jury in the first sample test. The string that specifies this test is "`RDRDR`".

(1,1)	(1,2)	(1,3)	(1,4)
(2,1)	(2,2)	(2,3)	(2,4)
(3,1)	(3,2)	(3,3)	(3,4)

Picture 1. The chosen path is "`RDRDR`".

The picture 2 shows the first query. The cells of the path "`DDRRR`" are denoted with bold frame. The cells that belong to the path chosen by the jury, the ones that have the detector that reports the presence of the robot, are shaded.

The picture 3 shows the second query. The cells of the path "`DRRRD`" are denoted with bold frame. The cells that belong to the path chosen by the jury, the ones that have the detector that reports the presence of the robot, are shaded.

(1,1)	(1,2)	(1,3)	(1,4)
(2,1)	(2,2)	(2,3)	(2,4)
(3,1)	(3,2)	(3,3)	(3,4)

Picture 2. Query "`DDRRR`".

(1,1)	(1,2)	(1,3)	(1,4)
(2,1)	(2,2)	(2,3)	(2,4)
(3,1)	(3,2)	(3,3)	(3,4)

Picture 3. Query "`DRRRD`".

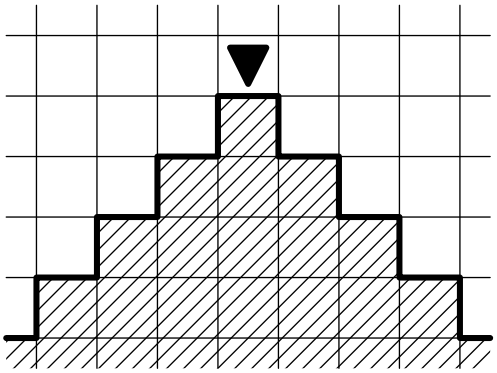
Problem E. Hide-and-Seek for Robots

Time limit: 1 second
Memory limit: 512 megabytes

Mike is creating robots that would play hide-and-seek game. The game of hide-and-seek for robots will take place on a rectangular grid that has the size of $m \times n$, some of its cells are occupied with robots.

Each robot looks in one of the four directions: upwards, downwards, to the left, or to the right. Each robot has its view area. Consider a robot that looks downwards. His view area contains one cell in the following row, three cells in the row next to the following, five in the next row, etc. The k-th row has a segment of $2k - 1$ cells in the robot's view area, its center is in the same column as the robot.

The view areas of the robots looking to the right, upwards and to the left are defined analogously.



The robot A sees the robot B if the cell where the robot B is located at is in the view area of the robot A. In order to start the hide-and-seek game for robots there must be no pair of robots on the playfield that both see another one from the pair. That is, for each pair A, B of robots at least one of two conditions must be satisfied: A is not in the view area of the robot B, or B is not in the view area of the robot A.

In one step Mike can turn any robot 90 degrees clockwise or counterclockwise. This operation changes the direction the robot is looking correspondingly. He would like to start the game as soon as possible, so he wants to make the minimum number of steps to get the configuration that allows to start the game.

Help him find such configuration. It can be proved that for any initial configuration of robots such configuration exists.

Input

The first line of input contains two integers m and n, the number of rows and the number of columns of the grid ($1 \le n, m \le 2\,000$).

The following m lines contain n characters each, each character is one of '`U`', '`D`', '`L`', '`R`', or '`.`'. The character '`U`' denotes the robot that looks upwards, the character '`D`' — the one that looks downwards, the character '`L`' — the one that looks to the left, the character '`R`' — the one that looks to the right, the character '`.`' means that this cell doesn't contain a robot.

Output

Output m lines, each must contain n characters — the configuration that allows to start the game. The configuration must be obtained from the one given in input by minimum number of steps. If there are several possible answers, you can output any one.

Note that Mike can only turn robots, he cannot remove them nor add them to the play field. So the robots in the output must be exactly at the same positions as in the input.

Examples

standard input	standard output
2 3 RDL .U.	UDL .R.
2 2	.. ..

Problem F. Isosceles triangles

Time limit: 1 second
Memory limit: 512 megabytes

You are given a regular polygon with n vertices. You should find the number of isosceles triangles with vertices that are the vertices of the polygon.

Input

The only line of input contains one integer n — the number of vertices of a regular polygon ($3 \le n \le 10^9$).

Output

The output line should contain a single integer — the number of isosceles triangles with vertices that are the vertices of the polygon.

Examples

standard input	standard output
3	1
5	10

Note

An isosceles triangle is a triangle that has at least two equal sides.

Problem G. Too Many Hyphens

Time limit:	1 second
Memory limit:	512 megabytes

Document markup languages such as TEX and LATEX, originally developed by Donald Knuth and Leslie Lamport, are often used in typesetting various articles, scientific texts, and even the statements for programming olympiads.

They provide special macros which allow one to use special characters that aren't present on the keyboard. For example, several hyphens in a row are converted into dashes of various lengths. Thus, when it's necessary to get a line of several hyphens in a document, you need to make some special actions.

In this case one can use braces to prevent a group of successive hyphens to be a macro. As in many programming languages (C++, for example) braces are used in TEX to group character blocks. Braces sequence should also form a correct parentheses sequence. That means that after removing all the characters from the line except braces, and then replacing the opening brace "`{`" with $+1$ and closing brace "`}`" with -1, the sum of all the elements of the resulting sequence should be equal to 0, and the sum of any prefix of the resulting sequence should be non-negative.

To prevent the sequence of hyphens from being replaced by a dash there should be at least one brace between any two consecutive hyphens. Note that the group of consecutive pluses, unlike hyphens, does not correspond to any macro and thus there are no extra conditions for two consecutive pluses.

Let's consider the string s consisting of pluses and hyphens. Let's add the minimum number of curly braces considering the rules described above, so that there are no two consecutive hyphens. Let's call the resulting string an *optimal escaped* string for s.

For example, for the string "`++--`" there are exactly five optimal escaped strings: "`++-{-}`", "`++-{}-`", "`++{-}-`", "`+{+-}-`", "`{++-}-`".

The lines above are listed in *lexicographical order*: they are ordered by the first unequal character (by the first, then by the second in case the first characters are equal, etc). Characters are ordered as "`+`" < "`-`" < "`{`" < "`}`".

Let's consider all the minimal length strings that can be obtained from s by adding bracers by the described rules, namely all optimal escaped strings for s. You must find the k-th in lexicographical order optimal escaped string or detect that the given k is too great.

Input

The first line of input contains the string s, consisting of pluses and hyphens. It's guaranteed that s is not empty and has a length of no more than 60 characters.

The second line contains a single integer k ($1 \le k \le 10^{18}$).

Output

You need to print the k-th in lexicographical order optimal escaped string for the string s specified in the input.

If k is greater than the number of optimal escaped strings, your program should print "`Overflow`".

Examples

standard input	standard output
++-- 2	++-{}-
-+-+- 2	Overflow

Problem H. Planet Nine

Time limit: 1 second
Memory limit: 512 megabytes

Planet Nine is a hypothetical planet in the outer region of the Solar System. Trying to prove the hypothesis of its existence, scientists have built a special device which is able to track astronomical objects. They have set it to observe the hypothetical orbit of the planet.

Scientists are really tight on budget, so the device has only one integer register. Every time an astronomical object is tracked by the device, the value in the register is increased by nine. The device has a special feature: the first digit in the decimal representation of the stored value may occasionally disappear, but only if it is equal to one. If the register stores a number 1, it can also disappear, leaving 0 as the new register value.

Let us name the addition of one or several nines — the operation of the first type, and disappearing of one or several leading digits one — the operation of the second type.

The register contained the value equal to a, and the scientists went for lunch. When they came back, they saw that the register has changed its value to b. They are wondering whether the device is broken, and, if not, they want to know the way of transforming a into b. The lunch was short, so such a transformation must be no longer than 1000 operations.

If there is more than one way to get b from a, the scientists will be satisfied with any of them.

Input

The first line of input contains two integer values a and b ($0 \le a, b \le 10^9$).

Output

If the device is broken, print "`Broken`" in a single line of output.

Otherwise the first line of output should contain the only word "`Stable`". The following lines should describe the way of getting b from a.

The second line of output should contain a single integer n ($0 \le n \le 1000$) — the number of operations that happened during the lunch. Note that it is not required to minimize n.

Each of the following n lines should describe an operation.

- Line "+ x", where $x > 0$, means that x astronomical objects were tracked by the device, and the register value was increased by $9x$.
- Line "- y", where $y > 0$, means that the first y digits disappeared from the register. All of those y digits must be equal to 1 to perform this operation.

After consecutive application of those n operations the value a must change to b. Temporary values achieved by applying the first k operations ($0 < k < n$) should not be greater than 10^{18}.

Examples

standard input	standard output
0 0	Stable 0
1 9	Stable 2 + 2 - 1

Problem I. Dates

Time limit: 1 second
Memory limit: 512 megabytes

One of the problems in dates processing is that there are several formats of writing down dates.

Peter has the task to write a program that processes an array of texts. Some of them have Russian and European format "day.month.year", while others use American format — "month/day/year".

Here, year is a number from 1 to 9999, which may contain leading zeroes to get 2, 3 or 4 digits, month — number from 1 to 12, which may contain leading zero to get 2 digits, day — number from 1 to 31, which may contain leading zeros to get 2 digits. Dates can be incorrect (for example there can be "Since 2001 isn't a leap year, that's why 29.02.2001 — incorrect date").

You've got an array of Peters' strings, which correspond to dates. Print each date in two formats: first as "`DD.MM.YYYY`', second as "`MM/DD/YYYY`".

Input

The first line of input contains one integer n ($1 \le n \le 20\,000$).

Each of the following n lines contain one date in format "`day.month.year`" or "`month/day/year`".

Output

Print n lines. Each line should contain the date in two formats: first as "`DD.MM.YYYY`", second as "`MM/DD/YYYY`".

Examples

standard input	standard output
2 11.12.2000 1.2.1	11.12.2000 12/11/2000 01.02.0001 02/01/0001
2 20.10.2100 1/29/3000	20.10.2100 10/20/2100 29.01.3000 01/29/3000

Problem J. Factory

Time limit:	2 seconds
Memory limit:	256 megabytes

It is suspected, that the military factory on the outskirts of the town is connected with a series of recent crimes. Public Safety Bureau sent inspector Akane to find out if that was the case.

Akane has a rectangular map of the factory, which is a table of size $m \times n$. Each cell of the map is either empty, or contains one of the factory's workshops. Of course, all workshops are cell-connected. That is, it's possible to get from any workshop to any other workshop through several intermediate workshops, going only between workshops sharing a side. It's also guaranteed, that the factory doesn't enclose any area that doesn't belong to the factory. That is, it's possible to get from any non-workshop cell to the border of the map, going only through non-workshop cells sharing a side.

Akane believes, that in case there are some clues to find, it's more likely that they are located in the corner of some workshop, rather than in it's center. For the above-mentioned map consider all grid nodes that are at the corner of at least one workshop. Akane calls such nodes important. It's easy to see, that the number of important nodes is at most $(m+1) \cdot (n+1)$. Akane also found out, that there are corridors on the border of each workshop cell connecting adjacent grid nodes at the corners of this cell.

Akane wants to traverse all the important nodes and to do it in a most efficient way. More precisely, Akane wants to make a route going through the nodes of the grid, such that:

- the route starts and ends at the same node;
- between every two nodes Akane goes through a single corridor;
- each important node is visited at least once;
- route goes through important nodes only;
- Akane goes through every corridor at most once.

Please help Akane find any valid route or determine, that it's impossible. Please note, that it's not required to minimize the length of the route.

Input

The first line contains integers m and n ($1 \le m, n \le 20$) — the size of the map with the factory.

The following m lines contain n characters each and describe the factory's map. The character "`*`" denotes the workshop, while "`.`" denotes an empty cell.

Output

In case it's impossible to traverse the factory, print "`No`".

Otherwise print "`Yes`" and then the number of corridors Akane will go through.

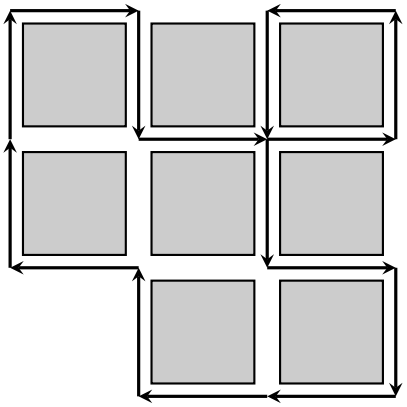
Then print the route itself, one grid node per line. Let's number all horizontal grid lines from 0 to m upside down and number all vertical grid lines from 0 to n from left to right. Each grid node is described with two integers r_i and c_i ($0 \le r_i \le m$, $0 \le c_i \le n$) — the row number and the column number.

Examples

standard input	standard output
3 3 *** *** .**	Yes 16 0 0 0 1 1 1 1 2 1 3 0 3 0 2 1 2 2 2 2 3 3 3 3 2 3 1 2 1 2 0 1 0
1 4 ****	Yes 10 0 0 0 1 0 2 0 3 0 4 1 4 1 3 1 2 1 1 1 0
2 2 ** **	No

Note

Akane's route in the first example is as follows:



Problem K. RotationAlmostSort

Time limit: 1 second
Memory limit: 512 megabytes

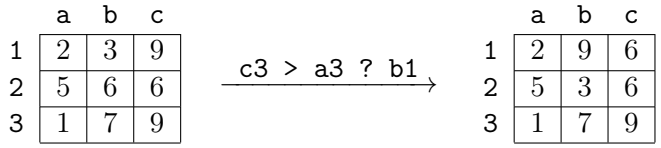
A robot can execute a sequence of commands in an $n \times n$ square filled with numbers. The only type of command the robot can execute is:

```
[cell 1] > [cell 2] ? [cell 3]
```

Such command has the following effect: if the number in cell 1 is greater than the number in cell 2, then rotate counterclockwise a 2×2 square whose upper left corner is cell 3.

Each cell is described with a letter denoting its column and a digit denoting its row, concatenated without a space. The columns are marked from left to right staring with `a`, the rows are marked from top to bottom starting with 1.

If one of the cells in one of the commands doesn't fit into the $n \times n$ square, or if cell 3 is located in the bottommost row or the rightmost column, then the robot becomes broken.



An example of a single command execution.
The condition is true ($9 > 1$), therefore the rotation is made.

After the execution of the entire program, the bottommost $n - 2$ rows are written down concatenated in the order from top to bottom (the 3-rd row, the 4-th row, ..., the n-th row). The resulting sequence consisting of $n \cdot (n - 2)$ numbers must be non-decreasing.

You are given an integer n. Output a program containing only commands of the described type, such that when a robot executes this program on any initial square with numbers, it doesn't become broken and makes the square satisfy the desired state.

Input

The first line contains a positive integer n ($2 \le n \le 9$) — the size of the square with numbers.

Output

Output a correct program. It must contain no more than 100 000 commands. Follow the output format shown in the example.

Example

standard input	standard output
2	a2 > b2 ? a1 a2 > b2 ? a1 a2 > b2 ? a1

Note

In order not to give a hint to the contestants, an example for $n \ge 3$ is not shown.

For $n = 2$ it is enough to obtain 0 non-decreasing rows, thus any syntactically correct program is a correct solution. But the shown program obtains not just 0, but 1 non-decreasing row in the bottom part of the square: it rotates the only 2×2 square counterclockwise until its bottom row becomes sorted non-decreasingly. It easy to prove that such situation will happen after 0, 1, 2 or 3 rotations.

Problem L. Time Travel

Time limit: 2 seconds
Memory limit: 512 megabytes

In future after the time machine was invented it became easier to learn history. Now you can just go to the corresponding year and watch the events by yourself. Professor uses this device to study the system of Berland roads during the Great Road Transformation.

The transformation took k consectuve years. During these years the system of roads of Berland used to change. There are n cities in the country, numbered from 1 to n. Each year these cities were connected by $n-1$ bidirectional roads, for every pair of cities there was a unique path between them.

Every day Professor chooses two cities s and f, travels to every year of the Great Road Transformation, one after another, and makes a trip from s to f in this year, visiting all roads on the path, including s and f. After that he wrote down the number of cities that were visited in all of his k trips.

Unfortunately, on the day he had completed his work, having had studied all pairs of cities, he lost all of his records. He only has maps of Berland road system for all k of the Great Road Transformation years.

Help Professor to restore the numbers that he had written down for all possible pairs of cities s and f.

Input

The first line of input contains two integers n and k — the number of cities in Berland and the number of years of the Great Road Transformation ($1 \le n, k \le 500$).

Then k descriptions of Berland roads systems follow, one for each year, in the following format: $n-1$ lines of the description contains two integers a and b each — the cities connected by the roads ($1 \le a, b \le n$, $a \ne b$).

It is guaranteed that for each of the k years for each pair of cities there is a unique path between them.

Output

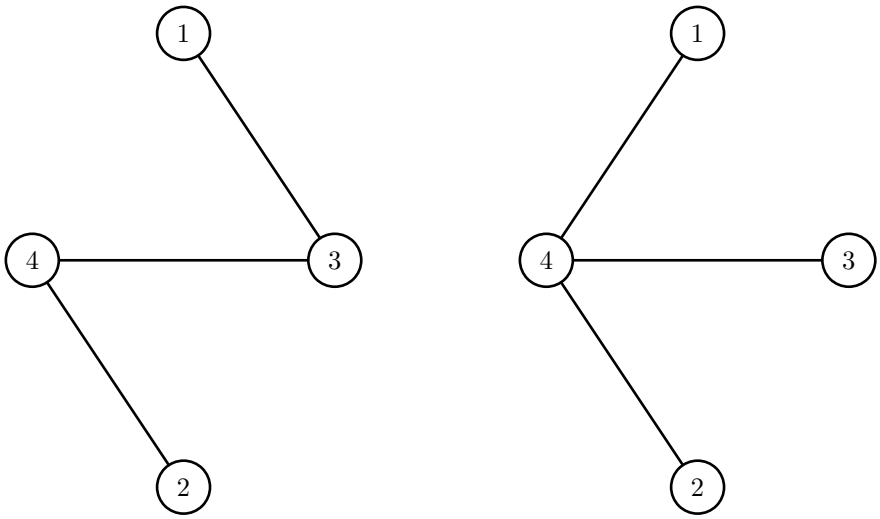
Output n lines, n integers in each of them. The j-th number in the i-th line must be the number that Professor wrote down if $s = i$ and $f = j$.

Examples

standard input	standard output
4 2 1 3 4 2 3 4 1 4 4 3 2 4	1 3 2 2 3 1 3 2 2 3 1 2 2 2 2 1
3 3 1 2 2 3 2 3 3 1 3 1 1 2	1 2 2 2 1 2 2 2 1

Note

There are 4 cities in Berland in the first sample test. There are 2 years studied by Professor. The map of the road system for the first year is shown on the left, the map for the second year is shown on the right.



Consider the pair of cities $s = 1$ and $f = 2$. Traveling to the first year, Professor visits cities 1, 2, 3, 4 on his trip. Traveling to the second year, Professor visits cities 1, 2, 4. So there are 3 cities 1, 2, 4 that will be visited in all his trips, so he writes down the number 3 for this pair of cities.

Consider the pair of cities $s = 3$ and $f = 1$. Traveling to the first year, Professor visits cities 1, 3 on his trip. Traveling to the second year, Professor visits cities 1, 3, 4. So there are 2 cities 1, 3 that will be visited in all his trips, so he writes down the number 2 for this pair of cities.