Problem A. Automaton

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

For a given string S, create acyclic determinstic finite-state automaton, that accepts all suffixes of S (and probably other finite strings). Automaton must consist of minimal number of states — N and no more than $2N$ transitions. Numeration of states is 1-based; without loss of generality we assume that all states are terminal.

Input

The only line contains string S, consisting of lowercase English letters ($1 \le |S| \le 10^5$).

Output

In the first line print two numbers N and K are amount of states and transitions, followed by K ($1 \le K \le 2N$) lines, each having two numbers a_i, b_i ($1 \le a_i, b_i \le N$) and lowercase English letter c_i, meaning transtion from state a_i to b_i by letter c_i. You can output transitions in any order. If there are multiple solutions, output any of them.

Example

standard input	standard output
abacaba	8 10 1 2 a 1 3 b 1 5 c 2 3 b 2 5 c 3 4 a 4 5 c 5 6 a 6 7 b 7 8 a

Problem B. Bank of a River

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

A couple decided to spend their vacation on a bank of a river. George likes high places and wants to go to the spot, where the bank is as high as possible above river level. But his wife Mary is afraid of heights and wants to spend the vacation where it's as low, as possible. Now they are going in a car along a one-way main road, and there are n turns to the river along it. The road over each turn leads to the river, and both spouses know the height of the spot, where corresponding road leads. Of course, George drives the car, but Mary can distract George, so he doesn't notice some of the turns (by her choice) except of the last one (and George knows about that). All the turns look so like each other, that George cannot be sure to which spot it leads. Main road continues over the last turn and leads to the spot with known bank height too. Obviously, George can use one of the following strategies: either he turns on first turn he noticed, on second, third etc., either not to turn at all (of course, in case if he notices less turns than required for his strategy, a couple will arrive at the place in the end of the main road).

Determine optimal strategy for George, supposing that Mary will act optimally too.

Input

The first line contains an integer N ($1 \le N \le 10^5$). The second line has $N+1$ integers h_i, ($0 \le h_i \le 1000$), which determine the height of the spot, where the road from i-th turn leads (h_{n+1} is height of the place, where the main road leads without any turns).

Output

Output maximum height of place, where George can get with optimal Mary's opposition in the first line. In the second line write $n+1$ number, which are probablities for George to use each of his clean strategies to reach the height. All the values have to be output with accuracy no less than 10^{-6}. In case if there are multiple optimal strategies, choose the one where probability of choice "Do not turn anywhere" is maximum. If there are multiple such strategies too, then choose the one with maximum probability of n-th turn choice, etc.

Example

standard input	standard output
2 0 6 3	4.000000 0.333333 0.666667 0.000000
3 2 3 4 2	2.800000 0.400000 0.400000 0.200000 0.000000

Problem C. Circles and Matrix

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

Consider matrix A of 1000 rows and 1000 columns.

Consider the *circle* with center in element $A_{i,j}$ and integer radius r as set of elements of $A_{x,y}$ such as $(i-x)^2 + (j-y)^2 \le r^2$.

You are given list of Q circles, numbered sequentially from 1 to Q Pair of circles a, b $(a < b)$ is called incorrect, if they have common cells and if there exists a circle c with $a < c \le b$ that isn't contained in the circle a.

Determine, if there is at least one incorrect pair of circles in the list.

Input

The first line contains an integer Q — length of the list ($1 \le Q \le 10^6$). Then Q lines are given, k-th of those lines describe one circle and contains three integers i_k, j_k and r_k — indices and radius of the circle ($0 \le r_k \le 500$, $1 + r_k \le i_k, j_k \le 1000 - r_k$. Note that circles in the list may coincide.

Output

If there is an incorrect pair of circles, write numbers of these circles (circles are numbered starting from 1 in order they are given in the input file). If there are more than one such a pair, print any of them. Circles in the pair may be printed in arbitrary order. If the given list does not contains an incorrect pair, print "`Ok`" instead.

Example

standard input	standard output
6 10 10 5 10 10 4 5 10 0 10 5 0 10 15 0 15 10 0	Ok
2 6 6 5 11 11 5	2 1

Problem D. Diagonals

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	256 mebibytes

You have simple polygon without self-intersections or self-tangency. You need to cut it diagonally into the minimum number of convex polygons. As the edges of convex polygons, you can use only the vertices of the original polygon. No three vertices of any convex polygon are collinear.

You can use sides of the original polygon and its diagonals as sides of new convex polygons y. Each of the convex polygons must lie within the original polygon. The union of all convex polygons must be the original polygon. Also there shouldn't be two convex polygons with a positive area of intersection.

Input

The first line contains the number N ($3 \le N \le 150$) — the number of vertices of the original polygon. Each of the next N lines contains two integers x_i and y_i ($-10\,000 \le x_i, y_i \le 10\,000$) — coordinates of i-th vertex of the polygon in counter-clockwise order. It is guaranteed that the polygon is non-degenerate.

Output

Output one number — minimum number of convex polygons the original polygon can be cut into.

Examples

standard input	standard output
7 3 -1 3 1 2 1 3 3 0 0 3 -3 2 -1	3

Problem E. Empire and Roads

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	256 mebibytes

The ancient Empire of Byteland had a road network designed for carriage transportation. Byteland was divided into several states. However, the division of the towns among the states was not fixed in time. Historians are now trying to recover the division at different points of time.

The only evidence the historians have about the examined period are chronicles telling which roads were classified as state roads at certain moments in history. The historians assume that the towns that were connected by a state road belonged to the same state. They are, however, uncertain if every pair of towns from the same state were connected by a state road and even if every pair of such towns were connected by a sequence of state roads (the connection could have used local roads which are not included in the historical data).

To test their methods, historians need a system which tracks the status of the roads in chronological order and answers the queries of the following form: at a given moment in history, given a set of towns, find out whether it is possible that the set of towns that belong to some single state is exactly the given set.

Input

The first line of the input contains two integers n and q ($1 \le n \le 1\,000\,000$, $1 \le q \le 2\,000\,000$), which specify the number of towns in Byteland and the number of events (road status changes according to the chronicles and historians' queries). The towns are numberedT from 1 to n.

Next q lines describe events in chronological order (a query corresponds to the moment in history when all road status changes above the query already took place, and all the following road status changes didn't). Each of these lines has one of the following formats:

- "1 u v" means that towns u and v are now connected by a state road ($1 \le u < v \le n$),
- "2 m" means that the road that became a state road during the m-th type 1 event is not a state road anymore (m is between 1 and the number of events of type 1 so far),
- "3 k u_1 u_2 ... u_k" represents a query whether the set of towns $\{u_1, u_2, \ldots, u_k\}$ could have formed a single state at the considered moment of time ($1 \le k \le n$, $1 \le u_1 < u_2 < \ldots < u_k \le n$).

Each road in Byteland is bidirectional. There is at most one road between any pair of different towns.

Initially there are no state roads. Any road may become a state road several times. No two events of type 2 share the same value of m. The sum of values of k over all queries does not exceed 2 000 000.

Output

For each query (event of type 3) in the input, your program should output one line with the word "YES" if the given set of towns could have been the full list of towns of some state at the respective moment in history, or the word "NO" otherwise.

Examples

standard input	standard output
4 10 3 3 1 3 4 1 1 2 3 3 1 3 4 1 2 3 3 2 1 3 3 3 1 2 3 1 3 4 1 2 4 2 1 3 3 2 3 4	YES NO NO YES YES

Problem F. Frogs

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

There is an infinitely long sequence of cells. For each $i \geq 0$, the beauty of the cell i is equal to $x^i \bmod p$.

Initially k smart frogs (numbered 1 through k) are standing at cell 0, and each of them has happiness equal to 1. They move according to the following steps:

1. Frog 1 moves one cell forward, and his happiness increases by the beauty of the cell he enters.
2. For each $i \geq 2$, if Frog $i-1$ moves and if the happiness of Frog $i-1$ is a multiple of m, Frog i will move one cell forward and his happiness increases by the beauty of the cell he enters. Otherwise Frog i does nothing. In this step, Frog 2 moves first, Frog 3 moves next, and so on.
3. If the distance between Frog 1 and Frog k is more than or equal to d, the movement ends. Otherwise, the frogs will repeat moves from step 1.

Compute the position of Frog 1 when they finish the movement.

Input

The input contains five integers x ($1 \leq x \leq p-1$), p ($2 \leq p \leq 10^5$), k ($2 \leq k \leq 10$), m ($2 \leq m \leq 10$), and d ($1 \leq d \leq 10^{12}$).

It is guaranteed that p **is a prime**.

Output

Output the position of Frog 1 when they finish the movement.

Examples

standard input	standard output
1 2 3 2 10	14
58 10007 10 10 123456789012	123456789143

Problem G. Graph Modifications

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

Given an undirected graph G. It satisfies the following properties:

- G is simple, that is, it contains no self loops or multiple edges.
- G is connected.
- G contains no simple cycles which have length at least 4. A tuple of k distinct vertices $v_1, \ldots, v_k$ is called a simple cycle of length k if for each i, v_i and v_{i+1} are connected by an edge, and additionally, v_1 and v_k are also connected by an edge.

Return the maximal number of edges you can add to G, while keeping the properties above. You can add an edge between any pair of vertices.

Input

The first line of the input contains two integers V ($1 \le V \le 10^5$) and E ($0 \le E \le 10^5$), separated by a single space. Here, V is the number of vertices of G, and E is the number of edges of G.

The following E lines are the descriptions of edges. The i-th of these lines contains two integers a_i and b_i ($1 \le a_i < b_i \le V$), separated by a single space. Here, a_i and b_i are endpoints of the i-th edge. Vertices are numbered 1 though V. It is guaranteed that G satisfies the properties in the statement.

Output

Output the maximal number of edges you can add to G.

Examples

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

Problem H. Hacked "Lines"

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

In this task, we will tell about the classic version of the game "Lines". There is a square field of 9×9 cells. Each cell can be empty or contain a single ball of one of 7 colors. Each turn the computer puts three balls of random colors in random empty cells. After that player can move one ball in any empty cell. If 5 or more balls of the same color arranged in a vertical , horizontal or diagonal lines, these balls are destroyed, and the player gets points and an extra turn.

In this task, everything is a bit different: we hacked the computer and are now able to select empty cells on each turn, which will be occupied by new balls, and colors of these new balls. Two new balls appear on each turn. Player doesn't move anything (because we are directing the actions of the computer). Our goal — to make such moves, that allow us on the first destruction to destroy as many balls as possible. That is, you can make a few moves to build some successful design, and then destroy it in one move . You cannot make additional destructions to clear map.

Two balls are put in the field simultaneously. Thereafter, any ball that belongs to the vertical, horizontal or diagonal line of at least 5 balls of the same color, instantly collapses. If there are less than two empty cells, the next turn is not available.

Input

Input contains exactly 9 rows by 9 characters each — the current position in the game. Each character is either a digit from 1 to 7 , meaning the color of the ball in the current cell, or character '.' if the cell is empty. It is guaranteed that there is no line with 5 or more balls of the same color.

Output

The first line of output should contain two numbers D and M — maximum number of balls that can be destroyed by one stroke and the number of moves that lead to this destruction. $2M$ lines follow(2 lines on each turn). Each line should contain three numbers r_i, c_i and $color_i$ ($1 \le r_i, c_i \le 9$ $1 \le color_i \le 7$) — number of row and column of empty cells, where ball of $color_i$ color was placed. Both ball of one turn were placed at the same time. At the time of first destruction you must destroy exactly D balls (theoretically its possible to destroy them in a few times and make extra moves, but this is not recommended).

If no one ball can be destroyed, D must be 0. M does not exceed 100.

Examples

standard input	standard output
723134552 2421.2456 353...442 14...2..4 23.2....4 4....1..1 1222..112 3621..124 631211777	25 9 6 7 2 3 4 2 5 8 2 5 7 2 5 3 2 6 3 2 3 6 2 8 5 2 7 5 2 6 5 2 5 5 2 2 5 2 3 5 2 4 8 2 4 7 2 4 4 2 5 6 2 4 5 2

Problem I. ICPC Abbreviatons

Input file:	*standard input*
Output file:	*standard output*
Time limit:	8 seconds
Memory limit:	512 mebibytes

Some universities have rather long names, so they abbreviate them when registering for ICPC programming competitions.

It turns out that the teams whose universities have lexicographically smaller names tend to sit closer to the buffet during the contest. Because of that, each school would prefer to use the lexicographically smallest possible abbreviation of its name.

Formally, we define a nonempty substring of the university's name that consists of letters and is not preceded nor followed by a letter as a *word*. To abbreviate the name, you are allowed to replace a suffix of each word with a dot ("."). However,

- you are not allowed to replace the whole word with a dot, and
- if you leave the word unabbreviated (the chosen suffix is empty), you should not add the dot.

It is allowed to not abbreviate any word and leave the university's name unchanged.

To compare abbreviated names lexicographically, remove all non-alphabetic characters, convert all the remaining characters to lowercase, and compare the resulting strings.

Input

The first line of input contains the university's name. It consists of letters of the English alphabet, spaces, commas (","") and hyphens ("-"). It is guaranteed that the input contains no two consecutive spaces and does not start nor end with a space, and also contains at least one letter. The number of characters in the given line, including spaces, is positive and will not exceed 10^6.

Output

Output the lexicographically smallest abbreviation of the university's name. Do not change the capitalization of the letters nor the positions of nonalphabetic characters. If there are many smallest abbreviations, any one of them will be accepted.

Examples

standard input	standard output
University of Warsaw	Uni. of W.
Byteland State University, Bytesburg	B. State U., B.
university of fence-building	uni. o. f.-b.
,Carnegie--,--MelloN-,- , -- University-	,Ca.--,--MelloN-,- , -- U.-
Udmurtian University	Udm. U.

Note

For example, "`Uni. of. W.`" is the smallest possible abbreviation for "`University of Warsaw`". This is because the string "`uniofw`" is lexicographically smaller than strings corresponding to all other possibilities, such as "`uow`", "`uniofwa`", or "`universityofwarsaw`".

Problem J. Jigsaw Puzzle

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

You got the new jigsaw puzzle for your Birthday. It is basically a box consisting of n rows and m columns. In each cell, there is a card with a positive integer. When all the cards are at their initial place, the box looks like this:

1	2	3	...	m
m+1	m+2	m+3	...	2m
2m+1	2m+2	2m+3	...	3m
...	...	...	...	...
(n-1)m+1	(n-1)m+2	(n-1)m+3	...	nm

Once you came home and saw that someone has played with your puzzle. Now It looks like this:

1	n+1	2n+1	...	(m-1)n+1
2	n+2	2n+2	...	(m-1)n+2
3	n+3	2n+3	...	(m-1)n+3
...	...	...	...	...
n	2n	3n	...	mn

You want to restore the initial state of the puzzle. To do this, you decide to perform a sequence of operations. Each operation consists of the following steps:

1) Raise any card up in the air
2) Find its correct position in the table
3) If this place is available, put the card that was in the air on it
4) Otherwise, swap the card lying at this place, with the card in the air, and return to step 2.

Additional restriction is that each of nm cards should appear in the air at least once. This means that even if the card is already on its place, you have to spend operation on a lifting and putting it back.

What is the minimum number of operations you have to perform to make your table look better?

Input

Single line contains two positive integers n and m, such that $n \cdot m \leq 10^{14}$.

Output

Print a single number - the minimum number of required operations.

Examples

standard input	standard output
4 4	10
2 3	3

Problem K. Kingdom Division

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

King Byteasar has ruled the Kingdom of Byteotia for several years. Nowadays he is not as strong and healthy as he used to be in the old days. That is why he decided to divide the kingdom into parts that he would give to his sons, the princes of Byteotia. Byteasar has a really large number of sons.

To avoid arguments about succession, Byteasar would like each of the parts to have different size. Also, due to their lack of experience in ruling, a number of youngest princes may not be assigned any part of the kingdom.

The Kingdom of Byteotia can be modeled as an $n \times m$ rectangle divided into $n \cdot m$ unit square fields. Each prince's part must be composed only of whole unit square fields and must be connected, that is, between any two unit square fields in the part there must exist a path composed of unit square fields in which every two consecutive fields are adjacent along an edge. Each unit square field must be assigned to exactly one prince's part.

Byteasar would like as many princes to be assigned non-empty parts of the kingdom as possible. Help him perform such a division.

Input

The input contains two integers n and m ($1 \le n, m \le 1\,000$), which specify the size of the rectangle.

Output

The first line of the output should contain one integer k: the maximum number of princes that can be assigned a part of the kingdom. This number should be followed by an example description of a division: n lines containing m characters from the set $\{\texttt{A}, \ldots, \texttt{Z}\}$ each. The parts of the kingdom are represented by maximal (that is, non-extendible in either side) connected parts composed of the same letter. Note that different parts may still be described by the same letter. The size of each part (that is, the number of letters in the part) must be different.

It is guaranteed that, for every possible answer, there exists a way to denote its parts by letters in the above format using no more than 26 letters.

Examples

standard input	standard output
4 5	5 ABCCE BBCCE DDDEE DDEEE
1 12	4 AABCCCCAAAAA
8 3	6 CCC CCC ACB ACB ABB ABB CCC ABB

Problem L. Lovely Numbers

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	256 mebibytes

The number N is called lovely if $\frac{\sigma(N)}{N} = \frac{A}{B}$ where $\sigma(N)$ is the sum of all divisors of N. For given A and B find all lovely numbers from 1 to 10^{14} inclusive.

Input

Two positive integers A and B ($1 \le A, B \le 100$, A is an odd number (suddenly!), $2 \le \frac{A}{B} \le 5$).

Output

The first line should contain the integer K — the number of lovely numbers. Each lovely number has to appear exactly once. Numbers can be printed in any order. It is guaranteed that K does not exceed 1000.

Examples

standard input	standard output
5 2	3 24 91963648 10200236032

Problem M. Matryoshkas

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

Matryoshkas are wooden dolls nested one inside the other. Each doll has its own external volume out_i and volume in_i of the empty space inside it. One doll can be embedded into another, if the external volume of the first one is smaller (here you ask yourself "smaller???" Yes, you were not confused. Exactly smaller) than the empty space inside the second. We can directly put no more than one doll inside the other. But, this nested doll can contain another doll and so on. That is, if the two dolls are actually located inside the third doll, than one of them is located inside the other.

Matryoshkas are nested optimally if the total volume of empty space inside all matryoshkas is minimal. Find the number of ways to optimally place matryoshkas into each other.

Input

The first line contains integer N ($1 \le N \le 100\,000$) — the number of matryoshkas in set. The i-th of the following N lines contains two integers out_i and in_i ($1 \le in_i < out_i \le 1\,000\,000\,000$) — external volume and the volume of empty space inside the i-th matryoshka.

Output

The number of optimal placements of nesting dolls modulo$10^9 + 7$.

Examples

standard input	standard output
3 5 4 4 2 3 2	1