Problem A. Arithmetic on a Board

Input file:	arithmetic.in
Output file:	arithmetic.out
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 mebibytes

Evgenia wrote n positive integers on a board. Aleksey can erase any two integers x and y, replacing them by either $x + y$ or $x \cdot y$ or $|x - y|$. Aleksey makes replacements until there is only one integer left. What is the minimal integer Aleksey can get after all replacements?

Input

The first line of input contains an integer n ($1 \le n \le 100\,000$). The second line contains a space-separated list of n integers a_1, a_2, ..., a_n which Evgenia initially wrote on the board ($1 \le a_i \le 30$).

Output

On the first line, print the minimal integer Aleksey can get.

Examples

arithmetic.in	arithmetic.out
2 1 2	1
3 1 2 3	0
4 16 2 3 4	2

Explanations

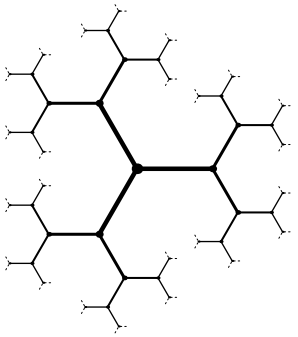
In the first example, you can replace 1 and 2 by $|2 - 1| = 1$.

In the second example, you can first replace 1 and 2 by $1 + 2 = 3$, and after that, get $|3 - 3| = 0$.

In the third example, one way to get a two is the following: $2 + 4 = 6$, $3 \cdot 6 = 18$ and $|16 - 18| = 2$.

Problem B. Covering Game

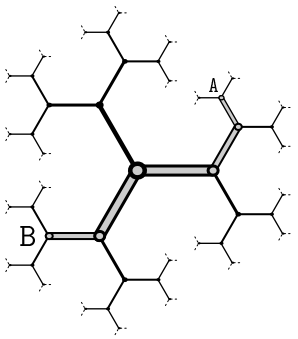
Input file:	`covering-game.in`
Output file:	`covering-game.out`
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 mebibytes



Picture 1

A *discrete trine* is similar to a discrete straight line consisting of points with integer coordinates. The difference is that from each point, one can move in three directions instead of two. A discrete trine could be conveniently represented on a plane as shown in Picture 1.

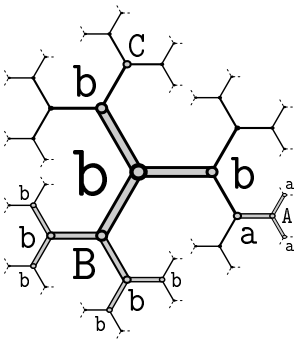
The *distance* on a discrete trine between points A and B is the minimal number of transitions from a point to its neighbour in order to move from A to B. For example, in Picture 2, the distance between points A and B is equal to 5.



Picture 2

A *circle* on a discrete trine with radius r and center T is a set consisting of all points of the trine which are no further than r from point T. Picture 3 shows the examples of three circles. Their centers are marked by capital letters, whereas other points of these circles are marked by the respective lowercase letters.

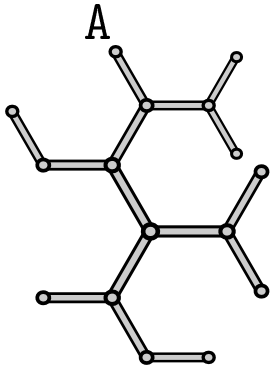
A *segment* of a discrete trine is much like a line segment: it is an arbitrary finite connected figure on a discrete trine. Such a segment can be interpreted as a graph drawn on the plane. This graph is in fact a tree, the degree of each vertex does not exceed three, and neighbouring edges form angles which are multiples of a 120-degree angle. An example segment is shown in Picture 4.



Picture 3

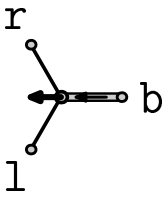
We will specify a segment by a *left-hand traversal* of the respective tree from one of its *leaves* which are vertices with only one neighbour in the segment. If there are no leaves in the tree, it consists of one vertex, and the left-hand traversal contains no transitions. In the general case, let us call the starting leaf the *root* of the tree. Now, for each vertex except the root, there is exactly one *ancestor*: the neighbour which is closest to the root of the tree.

The left-hand traversal is constructed recursively as follows. Assume we are at vertex v and look in the direction which we followed when moving to v from its ancestor (see Picture 5). If there is an edge from this vertex going forward and to the left, we move along that edge, recursively construct the traversal from the vertex we arrived to, and finally, move back to v. After that, if there is an edge from this vertex going forward and to the right, we similarly move along that edge, recursively construct the traversal from the vertex we arrived to, and finally, move back to v. As the direction is not defined for the root of the tree, we consider the only edge from the root to be the left edge.



Picture 4

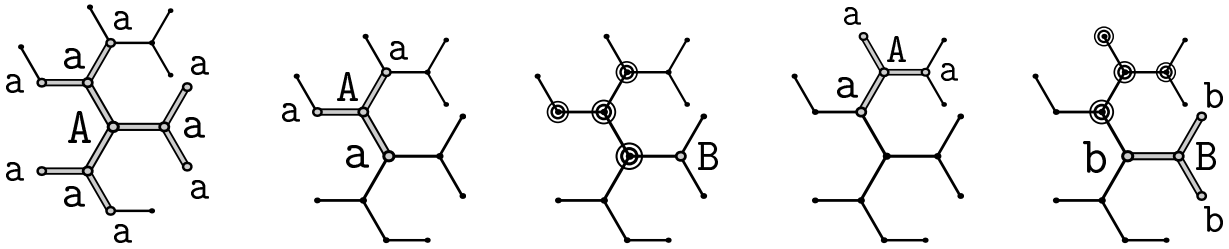
We will use character "`l`" to denote moving forward and to the left, character "`r`" for moving forward and to the right and character "`b`" for moving backwards to the root. For example, we will specify the segment shown in Picture 4. Let us start the traversal from the upper-most leaf marked by letter A. The segment can then be specified as follows: "`lllbrbbrlllbrbbrllbbrbbbbrrbbbb`".



Picture 5

Alice, Bob and Carl play a game on a discrete trine. Firstly, they choose the playing field: it is some segment of the discrete trine. The game ends when all points of the playing field are covered; initially, all points are considered not covered. The players make moves in turn: Alice moves first, Bob moves second, and Carl moves third. A move consists of choosing a point on the segment that is not yet covered. After that, a circle centered at that point is constructed. The radius of this circle is the maximal non-negative integer such that all points of the circle belong to the playing field and are not yet covered. Finally, all points of the circle are declared covered. The player who can not make the next turn loses the game.

Picture 6 shows some positions in the game and possible moves in these positions. The selected point is marked by a capital letter, and other points of the circle covered by the move are marked by respective lowercase letters. The double-circled points are the points covered earlier in the game. Remember that a player chooses the center of the circle, but after that, the radius can not be chosen: it must be the maximal possible for this center in the current game position.



Picture 6

Who of the players can be forced to lose? Player X can be forced to lose if the other two players can team up and act in such a way that X will have no way not to lose the game.

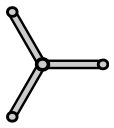
Input

The first line of input contains an integer m, the number of characters in the string which specifies a left-hand traversal of the playing field. The second line contains m characters without spaces: the left-hand traversal itself. It is guaranteed that this traversal correctly specifies a segment of a discrete trine, and this segment contains from 1 to 100 points.

Output

Print three lines. On the first line, print whether it is possible to force Alice to lose, on the second line, print the same for Bob, and on the third one, for Carl. Each line should consist of the word "`Yes`" in case of positive answer and "`No`" otherwise.

Example

covering-game.in	covering-game.out	Notes
6 llbrbb	No Yes No	

Explanation

In this simple example, Alice has only two considerably different moves: to choose one of the tree's leaves or the center vertex.

In the first case, only the chosen leaf will be covered, and each next move will also cover only one point. The total number of moves will be four. When there will be no possible moves left, it will be Bob's turn.

In the second case, all points are covered on the first move. Here, Bob also loses the game.

Problem C. Digit Statistics

Input file:	digit-statistics.in
Output file:	digit-statistics.out
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 mebibytes

Recently, while surfing the internet, Sergey read the following. If you take an arbitrary country of average size and then for each settlement write down its population, you can make an interesting observation. The amount of numbers starting with one will be greater than the amount of numbers starting with two. The latter amount, in turn, will be greater than the amount of numbers starting with three, and so on. The pseudo-scientific explanation was that the population distribution is more similar to the exponential distribution than to the uniform one.

Knowing that one should not easily trust what he reads on the internet, Sergey decided to do his own research. He wants to observe the first and last digits of a, a^2, a^3, ..., a^{n-1}, a^n for some fixed number a. For simplicity, he decided that a will be a small integer.

Please help Sergey in his research. Given a and n, find out how often each possible digit occurs as the first and the last digit of numbers from the above sequence.

Input

The first line of input contains two space-separated integers n and a ($1 \le n \le 10\,000\,000$, $2 \le a \le 9$).

Output

On the first line of output, print nine numbers: how many times the digits one, two, three, ..., nine occur as the first digit of the sequence a, a^2, a^3, ..., a^{n-1}, a^n. On the second line, print ten numbers: how many times the digits zero, one, two, three, ..., nine occur as the last digit of the same sequence.

Example

digit-statistics.in	digit-statistics.out
5 2	1 1 1 1 0 0 0 1 0 0 0 2 0 1 0 1 0 1 0

Explanation

In this example, the sequence consists of numbers 2, 4, 8, 16 and 32. Each of the digits one, two, three, four and eight occurs once as the first digit. Among the last digits, the digit two occurs twice, and each of the digits four, six and eight occurs once.

Problem D. Forbidden Triples

Input file:	`forbidden-triples.in`
Output file:	`forbidden-triples.out`
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 mebibytes

Consider triples (a, b, c) of integers in range $[0, n)$. We say that a triple is *forbidden* modulo n iff some of the numbers a, b, c equals the sum of others modulo n.

For example, triple $(5, 4, 3)$ is forbidden modulo 6 as $3 = (5 + 4) \bmod 6$.

Find the total count of forbidden triples modulo n.

Input

The only line of input contains a single integer n ($1 \le n \le 1\,000\,000$).

Output

Write the only line: the count of forbidden triples modulo n.

Examples

forbidden-triples.in	forbidden-triples.out
1	1
2	4

Explanations

In the first example, the only triple $(0, 0, 0)$ is forbidden as $0 = (0 + 0) \bmod 1$.

In the second example, the four forbidden triples are $(0, 0, 0)$, $(0, 1, 1)$, $(1, 0, 1)$ and $(1, 1, 0)$.

Problem E. Gun (Division 1 Only!)

Input file:	gun.in
Output file:	gun.out
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 mebibytes

In the University of Guns, people invent various new devices. Guns mostly. Fedor invented a gun that fires two bullets simultaneously: one like any other gun and another in the opposite direction. If one looks from above, the union of tracks of bullets looks like a straigt line.

Two years later, this weapon was already used in a real fight. According to the battle report, there were n enemy soldiers in different locations on the plane. Then, m shots were made, possibly from different points of the plane. For each shot, it is known which enemies were hit by this shot. Each shot hit from 1 to n enemy soldiers. Each enemy soldier was hit from 0 to m times.

Your task is to determine if this was possible. Note that the gun is very powerful, so it could easily hit all enemies in one shot if they stood in a straight line. You may also assume that nobody moved during the fight.

Input

The first line of input contains two integers n and m ($1 \le n, m \le 6$). The next m lines describe the shots. The first number on each of these lines is the number of enemy soldiers hit by the respective shot (it is an integer from 1 to n). Then follow the numbers of enemies hit by this shot without repetitions. For convenience, enemies are numbered by integers 1 through n.

Output

If the situation described in the input is possible, print "YES" on the first line. After that, print n lines each containing two integers in the range from 1 to 10 000: the coordinates of enemy soldiers. If there are several possible answers, you may output any one of them. It is guaranteed that if there exists an answer with arbitrary coordinates, there also exists an answer with coordinates satisfying the restrictions above.

In case the described situation is impossible, print "NO" on the first line. Remember that the locations of enemy soldiers must be different.

Examples

gun.in	gun.out
3 2 2 1 2 2 1 3	YES 1 1 1 2 2 1
3 2 3 1 2 3 2 1 3	NO

Explanations

In the first example, enemy soldiers can be placed in vertices of any non-degenerate triangle with coordinates satisfying the constraints.

The situation described in the second example is impossible: according to the information about the first shot, all three enemy soldiers stand on a straight line, but the second shot hit only two of them.

Problem F. Phi

Input file:	`phi.in`
Output file:	`phi.out`
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 mebibytes

This time, your task is very simple. Just calculate the sum of values of Euler's totient function between a and b, inclusive, i. e.

$$\sum_{i=a}^{b} \varphi(i).$$

Here, the Euler's totient function $\varphi(n)$ is the number of integers between 1 and n, inclusive, that are relatively prime to n.

Input

The input contains two integers a and b ($1 \le a \le b \le 4 \cdot 10^{12}$, $b - a \le 2 \cdot 10^6$).

Output

Output the value of the sum.

Example

phi.in	phi.out
2 4	5

Explanation

In the example, $\varphi(2) + \varphi(3) + \varphi(4) = 1 + 2 + 2 = 5$.

Problem G. Restoring Points (Division 1 Only!)

Input file:	`restore.in`
Output file:	`restore.out`
Time limit:	5 seconds (*7 seconds for Java*)
Memory limit:	256 mebibytes

Eva has marked 11 distinct points with integer coordinates on the plane. Then, she calculated the squared distance between each pair of distinct points. After that, she wrote down the 55 numbers she got in the order she wanted, and then erased the marked points. Finally, she showed the numbers to Andrew. Help Andrew restore the points on the plane.

Recall that the squared distance between points (x_1, y_1) and (x_2, y_2) on the plane is equal to

$$(x_2 - x_1)^2 + (y_2 - y_1)^2.$$

Input

The first line of input contains 55 integers separated by spaces: the pairwise squared distances between sought points. It is guaranteed that these are squared distances between 11 distinct points, and these points' coordinates are integers not exceeding 5 000 by absolute value.

Output

Print 11 lines containing the coordinates of points. Coordinates of a point are two integers: x-coordinate and y-coordinate. Coordinates of each point should be printed on a separate line and separated by a space. The collection of squared distances between the points you output should be the same as the collection of numbers in the input. You can output any answer in which coordinates of all points do not exceed 10 000 by absolute value.

Example

restore.in	restore.out
1 4 9 16 25 36 49 64 81 100 1 4 9 16 25 36 49 64 81 1 4 9 16 25 36 49 64 1 4 9 16 25 36 49 1 4 9 16 25 36 1 4 9 16 25 1 4 9 16 1 4 9 1 4 1	0 0 1 0 2 0 3 0 4 0 5 0 6 0 7 0 8 0 9 0 10 0

Explanation

In the example above, distances are given in the following order:
$(1, 2)$, $(1, 3)$, $(1, 4)$, ..., $(1, 11)$;
$(2, 3)$, $(2, 4)$, ..., $(2, 11)$;
...;
$(9, 10)$, $(9, 11)$;
$(10, 11)$.
All points lie on the same line.

The numbers in the example input occupy multiple lines only for readers' convenience. In fact, there is only one line with 55 numbers in each input.

Problem H. Robopatrol (Division 1 Only!)

Input file:	`robopatrol.in`
Output file:	`robopatrol.out`
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 mebibytes

Robopatrol is a patrolling robot. It moves along some cyclic route on a checked field. The robot transmits information about its movements. Unfortunately, sometimes errors are introduced by the transmission channel. Our task is to restore the exact route of the robot.

It is known that the route of the robot is a cyclic sequence of l cells without self-intersections in which any two consecutive cells share a side. The absence of self-intersections means that each cell is present in the sequence at most once. The length of the route is an integer between 4 and 200, however, the exact value of l is not given.

The patrolling route is transmitted by the robot as a sequence of directions of its movement on the field: "`D`" (downwards), "`L`" (to the left), "`R`" (to the right) and "`U`" (upwards). Each of these characters denotes moving in the neighbouring cell in the respective direction. It is known that, while moving along the route, the robot never makes two or more successive moves in the same direction.

The robot repeatedly moves along its cyclic route exactly n times and transmits the information about its movements in the order they take place. The number of repetitions is an integer between 10 and 20, however, the exact value of n is not given.

Unfortunately, the transmission channel used by the robot has an error probability p. It means that each transmitted character is distorted independently from other characters with probability p. The error probability is a real number between one and three percent and remains constant for the duration of transmission, however, the exact value of p is not given.

There are five possible types of distortion. The first three types change the direction character into some of the three other possible direction characters. The fourth type turns the direction character into character "`X`" which means the following: we know that the robot transmitted a character, but the character itself can not be recovered. The fifth type of distortion just skips the character completely. All five types of distortion are equiprobable.

Given the received data, restore the exact route of the robot. As the route is cyclic, you are allowed to restore it starting from any point of the route, and not just from the point which was the starting point for the robot.

Input

The first line of input contains an integer m. The second line contains m characters without spaces: the received data. It is guaranteed that the data satisfies the problem statement. The numbers n, l and p, as well as the route itself, are selected by the person preparing the input data, and the distortions are random.

Please keep in mind that, although the robot in fact does not move in the same direction two times in a row, the distortion may result in input data containing consecutive equal characters.

Output

On the first line, print l, the length of the cyclic route of the robot. On the second line, print l characters without spaces: the route itself in the form the robot would transmit it across an ideal (no distortion) channel as it moved along the route exactly once. You are allowed to start from any point of the route, but you must move along the route in the same direction. Please keep in mind that there is only one correct answer for each test case (not considering cyclic shifts): the exact route of the robot, and you must find this particular route.

Example

robopatrol.in	robopatrol.out
43 LURDLURDLURDLURDLURXLURDLURDLURDLURDLDDLURD	4 URDL

Explanation

In the example, $l = 4$, $n = 11$ and $p = 2.5\%$. During the transmission, 20-th character was changed into "X". The characters 38 and 39 also suffered from distortion: one of them was skipped, and another one was changed into "D".

Please note that the cyclic route of the robot is printed starting from the second character. The answers "LURD", "RDLU" and "DLUR" are also considered correct.

Problem I. AB-index (Division 2 Only!)

Input file:	abba.in
Output file:	abba.out
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 mebibytes

Consider string S of length N containing only letters 'a' and 'b'. An *ab-index* Z for this string is the number of distinct pairs of indices $1 \le i < j \le N$ such that $S_i = a$ and $S_j = b$. For example, $Z(abba) = 2$, $Z(abab) = 3$.

Find out the shortest string P such that $Z(P) = K$ for given K.

Input

The first line of input contains one integer K ($1 \leqslant K \leqslant 10^9$), the value of ab-index.

Output

Print the shortest string P containing only 'a' and 'b' such that $Z(P) = K$. If no such string exists, print "Impossible". In case of multiple solutions print any of them.

Examples

abba.in	abba.out
1	ab
3	abab

Problem J. Range Permutation Query (Division 2 Only!)

Input file:	rpq.in
Output file:	rpq.out
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 mebibytes

Given is a permutation P of N integers from 1 to N ($1 \le P_i \le N$, $P_i \ne P_j$ for any $1 \le i \ne j \le N$). Find the number of distinct ways to select a «subpermutation»: K sequential numbers $P_i, \ldots, P_{i+k-1}$ such that the resulting subsequence can be represented as a permutation of K integers from 1 to K.

For example, if $P_1 = 3$, $P_2 = 1$, $P_3 = 2$, the answer is 3 (for $K = 1$ and $K = 2$ we have subsequences starting with P_2, for $K = 3$ we have subsequence starting with P_1). If $P_1 = 2$, $P_2 = 3$ and $P_3 = 1$, the answer is 2 (for $K = 1$ starting with P_2, for $K = 3$ starting with P_1).

Input

The first line of input contains one integer N ($1 \le N \le 1\,000\,000$), the length of permutation. The second line contains N pairwise distinct integers P_i ($1 \le P_i \le N$).

Output

Print one integer: the number of ways to select a «subpermutation».

Example

rpq.in	rpq.out
3 2 3 1	2

Problem K. Triangles (Division 2 Only!)

Input file:	`triangles.in`
Output file:	`triangles.out`
Time limit:	2 seconds (*3 seconds for Java*)
Memory limit:	256 mebibytes

For two non-degenerate segments AB and CD, find a point E such that the area of triangle EAB will be equal to the area of triangle ECD. Each of the segments AB and CD is parallel to one of coordinate axes, and segments cannot have common inner points.

Input

The first line of input contains four integers x_A, y_A, x_B, y_B: coordinates of points A and B. Second line contains another four integers x_C, y_C, x_D, y_D: coordinates of points C and D. All coordinates do not exceed 1000 by absolute value.

It is guaranteed that each segment is parallel to one of the axes, i. e. either $x_A = x_B$ or $y_A = y_B$ (similar for C and D). It is also guaranteed that segments are non-degenerate and do not have an intersection point inside any of them (but one or both of points A and B may coincide with points C or D).

Output

If such a point E exists and its coordinates do not exceed 5000 by absolute value, print "`YES`" on the first line. In this case, on the second line, print coordinates x_E and y_E of such a point. Print coordinates with maximal possible precision. Difference in areas of triangles should not be greater than 10^{-3}. Area of each triangle should not be less than 0.1. If multiple solutions exist, print any one of them.

If such a point with coordinates not exceeding 5000 by absolute value does not exist, print "`NO`" on the first line.

Examples

triangles.in	triangles.out
0 0 0 4 5 2 5 3	YES 1.00000000000000 0.0000000000000000
1 0 3 0 1 0 1 1	YES 3.00000000000 1.000000000000000