

Problem A. Arithmetic on a Board

Input file: *standard input*
Output file: *standard output*
Time limit: 2 seconds
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 ($1 \leq n \leq 100\,000$). The second line contains a space-separated list of n integers a_1, a_2, \dots, a_n which Evgenia initially wrote on the board ($1 \leq a_i \leq 30$).

Output

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

Examples

standard input	standard output
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. Binary Tree

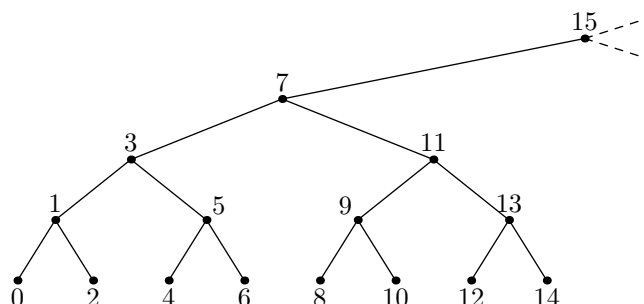
Input file: *standard input*
Output file: *standard output*
Time limit: 2 seconds
Memory limit: 256 mebibytes

Never underestimate the power of two!

quote of **7ania7** on
www.topcoder.com

In Speciality of Interval Trees and Heap (SITH) everyone knows that it is easy to enumerate vertices from top to bottom: root has a number of 1, its children are 2 and 3 and so on. It is really easy: you have to only divide vertex number by two to get its father's number.

But, as you know, everything is backwards in mathematics, and in this problem vertices are numbered from left to the right (see figure). Your task is very simple: you have to find sum of all numbers on vertices on a simple path from vertex a to vertex b . You can assume that root has a number of 55 213 970 774 324 510 299 478 046 898 216 203 619 608 871 777 363 092 441 300 193 790 394 367.



Input

First and only line of the input file will have two vertex numbers: a and b ($0 \leq a, b \leq 10^{15}$).

Output

The only line of output file should contain one number, the answer to the problem.

Examples

standard input	standard output
1 5	9
3 4	12

Problem C. Codeforces

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

Several years ago Vasya liked to compete in Codeforces competitions.

At the beginning of Codeforces era in each competition, participants were given a set of at most five problems. For each correct solution, contestants get some score depending on the time spent to solve the problem. Also there are other ways to earn or lose score. Needless to say that sometimes these ways become critical to the competition outcome.

For example, when all the problems are very tricky, the winner can have no problems solved. And when they are easy, even the last one on the scoreboard may have all the problems solved. Vasya thinks such problemsets are bad.

To formalize his thoughts, he invented a formula to calculate the badness of a contest. He assumes all contestants get distinct score, so there are no coders tied for the same place.

Let n be the number of competitors. Then Vasya defines the badness as

$$B = \frac{2 \sum_{i=1}^{n-1} \sum_{j=i+1}^n f(i, j)^2}{n(n-1)}.$$

Here, $f(i, j)$ is equal to zero if the i -th ranked coder solved more problems than the j -th ranked coder. Otherwise, $f(i, j)$ is the difference between the number of problems solved by i -th ranked coder and the number of problems solved by j -th ranked coder.

Input

Input consists of one or more test cases.

Each case starts with a line containing a single integer n ($2 \leq n \leq 100\,000$). Second line contains n integers — the number of problems solved by coders (from the first ranked coder to the last one; remember that there are no more than five problems in a competition). Input will be terminated with a test case with $n = 0$ which should not be processed. Sum of all n in the input doesn't exceed 100 000.

Output

For each test case write a single line with the answer. The answer can be integer or rational. In case of rational answer, numerator and denominator should be coprime, and denominator should be positive. Adhere to the sample output format below as close as possible.

Example

standard input
3
3 2 1
2
0 5
4
0 1 0 0
0
standard output
Case #1: The contest badness is 0.
Case #2: The contest badness is 25.
Case #3: The contest badness is 1/6.

Problem D. Digit Statistics

Input file: *standard input*
Output file: *standard output*
Time limit: 2 seconds
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 $a, a^2, a^3, \dots, a^{n-1}, a^n$ 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 ($1 \leq n \leq 10\,000\,000$, $2 \leq a \leq 9$).

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 $a, a^2, a^3, \dots, a^{n-1}, a^n$. 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

standard input	standard output
5 2	1 1 1 1 0 0 0 1 0 0 0 2 0 1 0 1 0 1 0

Example 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 E. Euclid

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

Vasya works for RIA (Research Institute of Archaeology). During recent excavations around Alexandria, Vasya found an interesting paper. He suggested it to be a game log between two Euclid's scholars. Euclid selected some integer number X . Then the scholars started writing out *euclid-maximal* pairs of natural numbers with respect to X . The scholar who could not write out any new pair lost the game.

We call a pair (a_m, b_m) *euclid-maximal* with respect to X if $1 \leq a_m \leq b_m \leq X$ and Euclid's algorithm for finding the greatest common divisor makes maximal possible number of steps among all pairs of integers (a, b) such that $1 \leq a \leq b \leq X$. The number of steps is the final value of variable **step** in the following pseudocode:

```
function gcd (a, b):  
|   step := 0  
|   while a > 0 and b > 0:  
|       |   step := step + 1  
|       |   if a >= b: a := a mod b  
|       |   else:     b := b mod a  
|   result := a + b
```

Unfortunately, the log was unfinished. Now Vasya needs a program to calculate the maximal possible length of the log. Obviously, that length is equal to the total number of euclid-maximal pairs with respect to given X .

Input

The input consists of one or more test cases. Each test case consists of a single line with a single integer X — the limit for numbers a and b ($1 \leq X \leq 10^9$).

The input will be terminated with a test case with $X = 0$, which should not be processed.

The total number of test cases will not exceed 10 000.

Output

For each test case write a single line with a single integer — the number of different pairs.

Example

standard input	standard output
1	1
2	3
3	1
4	2
5	1
10	1
0	

Problem F. Forbidden Triples

Input file: *standard input*
Output file: *standard output*
Time limit: 2 seconds
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 ($1 \leq n \leq 1\,000\,000$).

Output

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

Examples

standard input	standard output
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 G. Grasshopper

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

The Grasshopper lives on a one-dimensional meadow which is a set of points on a coordinate axis: the meadow consists of all points with integer coordinates from -10^{100} to $+10^{100}$, inclusive.

The Grasshopper has three pairs of legs. If the Grasshopper is located at point x , the first pair of legs allows him to jump to point $x + a$, the second pair to point $x + b$, and the third one to point $x + c$. An additional restriction is that the Grasshopper can not jump outside the meadow.

The Grasshopper is an optimist, so he always looks on the side where the coordinates of points increase. So, if the Grasshopper can jump from x to $x + t$, it does not automatically imply that he can also jump from x to $x - t$.

Initially, the Grasshopper is located at point 0. In order to reach some point of the meadow, he can jump any non-negative number of times using each of his three pairs of legs. The jumps by the means of different pairs of legs can occur in any order.

Calculate the approximate portion of points that can be reached by the Grasshopper, that is, the ratio of the number of points he can reach to the total number of points.

Input

The first line of input contains three integers a , b and c separated by spaces: the parameters of the three pairs of Grasshopper's legs ($-10^6 \leq a, b, c \leq 10^6$).

Output

On the first line, print one real number: the approximate portion of points that can be reached by the Grasshopper. The answer is considered correct if it differs from the exact answer by no more than 10^{-9} in absolute value.

Examples

standard input	standard output
2 2 2	0.25
0 -3 2	1.0

Explanations

In the first example, each of the three pairs of legs allows the Grasshopper to reach the point $x + 2$ from any point x . So, he can reach any point with noonegative even coordinate and can not reach any other point. The portion of such points is

$$\frac{5 \cdot 10^{99} + 1}{2 \cdot 10^{100} + 1} \approx 0.25.$$

In the second example, the Grasshopper can move from x to $x - 1$ jumping by $+2$ and -3 in some order. Moreover, he can move from x to $x + 1$ jumping by $+2$, $+2$ and -3 in some order. By repeating one of these sequences a certain number of times, the Grasshopper can reach every point of the meadow.

Problem H. Hand Gun

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

In the University of Guns, people invent various new devices. Guns mostly. Fedor invented a hand 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 straight 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 ($1 \leq n, m \leq 6$). 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 in order of their numbers. 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

standard input	standard output
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 I. Intersect the Lists

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

RIGS has another task for Vasya. Given are n lists of strictly increasing integers. His task is to write a program that will find the size of intersection of these lists, that is, the number of integers that are present in each of the lists.

Vasya knows that his program will be tested on generated lists. Each list contains exactly n integers $(y_0, y_1, \dots, y_{n-1})$ and depends on five given parameters (x_0, y_0, a, b, m) . The elements y_1, y_2, \dots along with auxiliary values x_1, x_2, \dots are generated as follows:

$$\begin{aligned} x_i &= (a \cdot x_{i-1} + b) \bmod 4\,294\,967\,296, \\ y_i &= y_{i-1} + 1 + (x_i \gg m). \end{aligned}$$

Here, $u \gg v$ means shifting the binary representation of u by v bits to the right. That is effectively the same as performing an integer division of u by 2^v . If m is greater than 31, the value of $x_i \gg m$ is equal to zero.

Input

Input consists of one or more test cases. Each case starts with a single line containing one integer n ($1 \leq n \leq 8\,000$). Each of the next n lines contains five integers x_0, y_0, a, b, m ($0 \leq x_0, y_0, a, b, m < 4\,294\,967\,296$) which are the parameters of the list. Input will be terminated with a test case with $n = 0$ which should not be processed.

The sum of n in all cases will not exceed 8 000.

Output

For each test case, write a single line with the answer to the problem. Adhere to the sample output format below as close as possible.

Example

standard input
3
0 0 1 1 0
1 0 30 239 1000000
0 0 0 2 1
2
0 0 2 5 1
1 1 5 2 1
0
standard output
Case #1: Intersection contains 2 integer(s).
Case #2: Intersection contains 0 integer(s).

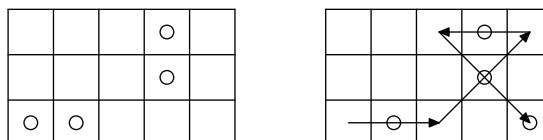
Problem J. Jumping Fleas and a Dog

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

Vasya got a new assignment: he has to write a simulator for the game “Jumping Fleas and a Dog”.

There are n fleas and one sleeping dog on an infinite chessboard. There can be more than one flea on a single cell. A move consists in choosing one flea and either moving it to a neighbouring cell or making one or several jumps with it.

Cells (x_1, y_1) (x_2, y_2) are said to be neighbouring iff $\max(|x_1 - x_2|, |y_1 - y_2|) = 1$. A jump can be made from cell (x_1, y_1) to cell $(2x_2 - x_1, 2y_2 - y_1)$, if cell (x_2, y_2) is occupied and is a neighbour of (x_1, y_1) , and cell $(2x_2 - x_1, 2y_2 - y_1)$ is free. See figure for examples of possible jumps.



The goal of the game is to reach the cell with the dog with one of the fleas.

Your task is to determine the minimal number of moves required to achieve the goal.

Input

The input consists of one or more test cases. Each test case starts with a single line containing three integer numbers n, x_d, y_d — the number of fleas and the position of the dog ($1 \leq n \leq 100\,000$). The following n lines contain pairs of integers — the coordinates of fleas. The input will be terminated with a test case with $n = x_d = y_d = 0$. This test case should not be processed.

Sum of n in the whole input will not exceed 100 000. All coordinates are integer numbers not exceeding 10^9 by absolute value.

Output

For each test case, write a single integer in a single line — the minimal number of moves required to achieve the goal.

Examples

standard input	standard output
4 0 -2 0 0 1 0 3 1 3 2 1 0 0 0 0 0 0 0	2 0

Problem K. Keywords

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

... where α is n -th letter of Latin alphabet

from a problem statement

Vasya works for RICKA (Research Institute of Cryptographic Keywords Assignment). He has written a number of programs that generate secret keywords based on users' personal data. However, these keywords are not always unique. To avoid this problem, Vasya decided to append some strings to each of the keywords, making all of them different.

To make the keywords closer to initial ones, he wants to keep the maximal length of an appended string as small as possible.

All keywords, both initial and resulting, must consist of letters 'a' and 'b'.

Input

The input consists of one or more test cases. Each test case starts with a line containing a single integer n — the number of keywords. The following n lines contain non-empty strings consisting of letters 'a' and 'b' — the initial keywords.

The input will be terminated with a test case with $n = 0$, which should not be processed.

The sum of lengths of all keywords in the whole input doesn't exceed $5 \cdot 10^5$.

Output

For each test case, write a single line containing the smallest possible maximal length of an appended string.

Example

standard input	standard output
2	1
a	0
a	2
2	
a	
b	
5	
a	
a	
a	
a	
ab	
0	

Problem L. Language Troubles

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

During his work at the Research Institute of Given Strings, Vasya faced huge problems, as the program for password hashing had a source in a language he didn't know.

```
[define[reversed_range n][if [< n 1] '[] [cons n [reversed_range [- n 1]]]]]
[define [range n] [reverse [reversed_range n]]]
; [range n] is a list of integers from 1 to n
[define [f l]
  [if [null? [cdr l]]
    [car l]
    [f [append [cdr[cdr l]] [list [car l]]]]]
]
[define [hash n] [f [range n]]]
```

Vasya is in a great necessity of being able to calculate `[hash n]`. Help him to deal with *Awful* programming language. You may assume that the interpreter has enough stack memory to calculate `[hash n]` for every n .

<i>Expression</i>	<i>Result</i>	<i>Expression</i>	<i>Result</i>
<code>[- 10 1]</code>	9	<code>[list 1]</code>	<code>[1]</code>
<code>[car '[1 2 3]]</code>	1	<code>[cdr '[1]]</code>	<code>[]</code>
<code>[cdr '[1 2 3]]</code>	<code>[2 3]</code>	<code>[cons 1 '[2 3]]</code>	<code>[1 2 3]</code>
<code>[append '[1 2] '[3 4]]</code>	<code>[1 2 3 4]</code>	<code>[reverse[range 5]]</code>	<code>[5 4 3 2 1]</code>
<code>[cdr[cdr[range 5]]]</code>	<code>[3 4 5]</code>	<code>[if [< 0 1] 1 0]</code>	1
<code>[if [null? '[1]] 1 0]</code>	0	<code>[if [null? '[]] 1 0]</code>	1
<code>[range 5]</code>	<code>[1 2 3 4 5]</code>	<code>[reversed_range 5]</code>	<code>[5 4 3 2 1]</code>

Input

The input consists of no more than a thousand test cases. Each test case consists of a single line with a single positive integer n ($1 \leq n \leq 10^{100}$). Input is terminated with a test case where $n = 0$. This test case should not be processed.

Output

For each test case, write a single line with a single integer — the value of `[hash n]`.

Examples

standard input	standard output
1	1
2	1
3	3
0	

Problem M. Multiple Barcodes

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

Vasya works for RIBCA (Research Institute of BarCodes Analysis). He is developing a new noise-proof barcode analysis system. The only disadvantage of the system is that it is limited with barcodes that are created in the following way.

Consider a grid $w \times h$ of black and white cells. Initially, all cells are black. Select no more than n rectangles with sides aligned with the grid lines. After that, for each selected rectangle, take all cells inside and invert them (white turns into black and vice versa).

Now Vasya wants to know the number of different barcodes that his analysis system can handle. Write a program to calculate it.

Input

The input consists of one or more test cases. Each test case consists of a single line containing three integers w , h and n — the size of the grid and the maximal number of rectangles ($0 \leq n \leq 2$, $1 \leq w, h \leq 10$).

The input will be terminated with a test case with $w = h = n = 0$ which should not be processed.

The total number of test cases will not exceed 10.

Output

For each test case write a single line with a single integer — the number of different barcodes.

Example

standard input	standard output
2 2 0	1
2 2 1	10
2 2 2	16
0 0 0	

Problem N. New Bus Routes

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

Vasya works for RIBBeRy (Research Institute of Buses and Bus Routes). He is writing a program that plans new routes. The problem he is solving at the moment is the following: given a set of places where route endpoints might be placed, find the number of suitable pairs of endpoints. A suitable pair is a pair for which a suitable route exists; a suitable route is any shortest route with length d within a fixed range ($l_i \leq d \leq r_i$). As the routes are intended to be placed in a city, the distance between two points is the sum of differences of their coordinates.

Help Vasya to write a program that finds the described number of pairs for a given set of endpoints and a set of ranges.

Input

The input consists of one or more test cases. Each test case starts with a line containing three integers n , k and z — the number of endpoints, the number of ranges and the size of the city ($1 \leq n \leq 5\,000$, $1 \leq k \leq 100\,000$, $1 \leq z \leq 1\,000\,000$). The following n lines contain pairs of integers (x_i, y_i) — coordinates of endpoints ($0 \leq x_i, y_i \leq z$). The next k lines contain pairs of integers (l_i, r_i) — ranges of distances to be checked ($1 \leq l_i \leq r_i \leq 2 \cdot z$).

The input will be terminated with a test case with $n = k = z = 0$, which should not be processed.

The sum of n 's in the whole input doesn't exceed 5 000. The sum of k 's in the whole input doesn't exceed 100 000. The sum of z 's in the whole input doesn't exceed 1 000 000.

Output

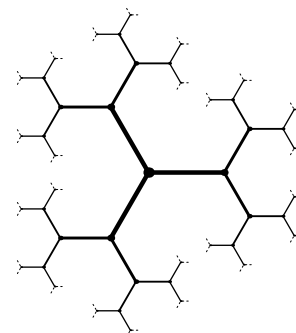
For each test case write k lines containing the number of pairs of points for each range. Output for each pair of consecutive test cases should be separated by an empty line.

Example

standard input	standard output
3 3 3	3
0 0	1
0 3	0
1 2	
1 3	0
1 2	0
1 1	
2 2 2	
0 2	
0 2	
1 4	
1 4	
0 0 0	

Problem O. Optimal Cheating

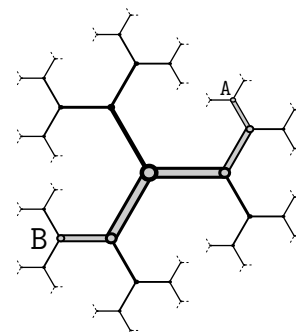
Input file:	<i>standard input</i>
Output file:	<i>standard output</i>
Time limit:	2 seconds
Memory limit:	256 mebibytes



Picture 1

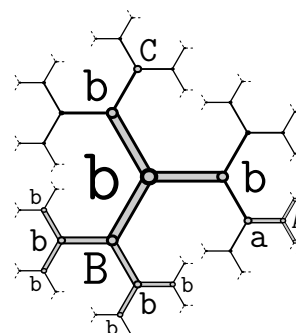
A *discrete trine* is similar to a discrete straight line consisting of points with integer coordinates. The difference is that from each point, one can move in three directions instead of two. A discrete trine could be conveniently represented on a plane as shown in Picture 1.

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.



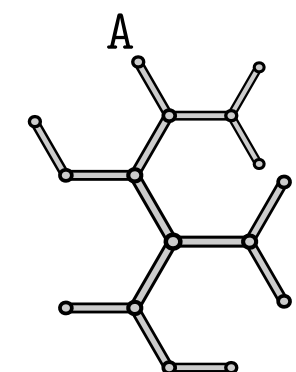
Picture 2

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.



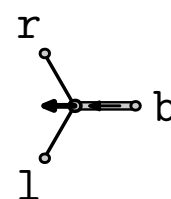
Picture 3

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 4

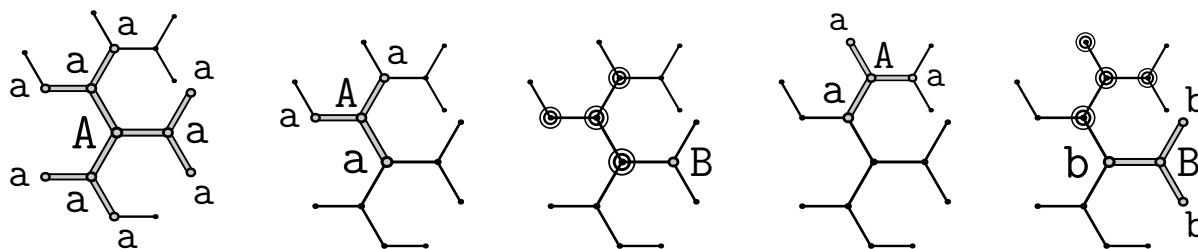
We will specify a segment by a *left-hand traversal* of the respective tree from one of its *leaves* which are vertices with only one neighbour in the segment. If there are no leaves in the tree, it consists of one vertex, and the left-hand traversal contains no transitions. In the general case, let 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.



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.

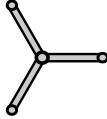
Input

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

standard input	standard output	Notes
6 11brbb	No Yes No	

Example 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 P. Phi

Input file: *standard input*
Output file: *standard output*
Time limit: 2 seconds
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 ($1 \leq a \leq b \leq 4 \cdot 10^{12}$, $b - a \leq 2 \cdot 10^6$).

Output

Output the value of the sum.

Example

standard input	standard output
2 4	5

Example explanation

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

Problem Q. Qualification Round

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

There are n sportsmen taking part in qualification round of major sports competitions held by Independent Department of Different Qualitative Defences (IDDQD). On the first day, they all stand in one line.

Too bad! Sportsmen should stand in the order of non-increasing height, and now they stand in some random order: the first one (from left) has height a_1 , the second one has a_2 , and so on. Nevertheless, there is a solution. No need to rearrange the sportsmen! Instead, they can be divided (virtually) into *several* lines.

For example, if four sportsmen with heights of 176, 174, 178, 168 centimeters stand in that order, one can say that there are two virtual lines with two sportsmen in each. And in this case, the heights of sportsmen in each of these lines will be in non-increasing order—just as planned!

Your task is to help organizers to calculate the number of such partitions. Every line in each partition must consist of several (at least one) sportsmen. Lines mustn't be mixed: if there are two sportsmen from one line, all sportsmen between them must be in the same line. In every line, sportsmen heights must be in non-increasing order from left to right. Every sportsman must belong to exactly one line.

Input

First line of input file will contain a non-negative integer n , the number of sportsmen ($n \leq 100\,000$). Second line will contain n integers a_1, a_2, \dots, a_n : heights of sportsmen ($1 \leq a_i \leq 10^9$).

Output

The only line of output file should contain one integer, the answer to the problem.

Example

standard input	standard output
4 176 174 178 168	4

Problem R. Regular Convex Polyhedra for Dummies

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

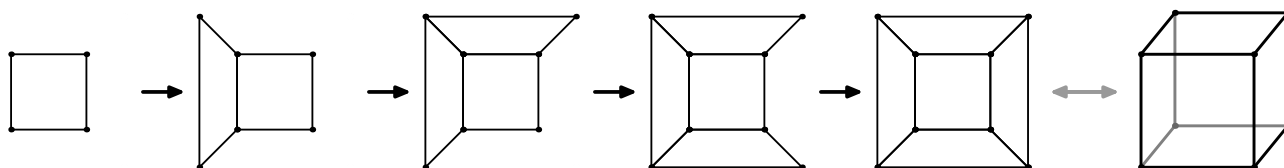
Vasya works for RID (Research Institute for Dummies). He develops new advanced techniques of teaching dummies. His current field of study is geometry for dummies, and the aim is to make complex geometric concepts as simple as possible. Vasya recently finished his work titled “Polygons for Dummies”, and is now looking forward to more advanced topics. Unfortunately, his research shows that most dummies have poorly developed three-dimensional imagination, so teaching them 3-D geometry will likely turn into a challenge.

Vasya decided to start the course by introducing the graphically appealing concept of regular convex polyhedra. Recall that a *regular convex polyhedron* is a convex polyhedron with all faces being equal regular polygons. In order to make things simple, Vasya makes a few preliminary observations before actually showing all regular convex polyhedra.

First observation is that each regular convex polyhedron can be characterized by two numbers: p — the number of vertices on each face and q — the number of faces coinciding in each vertex (this number is the same for every vertex of a regular convex polyhedron). For example, a cube has square faces, so $p = 4$, and each vertex of a cube has exactly three faces coinciding in it, so $q = 3$.

Another observation Vasya gives is the fact that the sum of polygon angles at each vertex of a polyhedron must be strictly less than 360° . For example, there is no regular polyhedron with $p = 6$ and $q = 4$, because a regular hexagon has an angle of 120° , so the sum of polygon angles in a vertex would be $120 \cdot 4 = 480^\circ$.

The last observation is that one can easily determine the number of vertices, edges and faces of a regular convex polyhedron without actually working in three dimensions. Suppose we know p and q ; draw a graph on the plane starting with a single p -gon. Try adding other p -gons in such a way that all vertices have degree not more than q , and as many vertices as possible have degree of exactly q . This process is illustrated below for $p = 4$ and $q = 3$.



If a polyhedron with the given p and q exists, eventually you will get a graph with all vertices having degree q ; all polyhedron's faces were added as p -gons except the last one which is the outer area of the plane — it turns out to have exactly p vertices as well. Note that since we draw the graph on a plane, not in three dimensions, the faces need not be regular p -gons (they even need not be convex). Still, the number of vertices, edges and faces in this planar graph is the same as in the polyhedron.

Vasya claims that these observations are enough to determine for each pair p, q whether a regular convex polyhedron with such characteristics exists, and if it does, how many vertices, edges and faces it has. Write a program which would do that for him.

Input

The input consists of one or more test cases. Each test case consists of a single line with two integers p and q on it — the number of vertices of a regular polygon and the number of polygon intersecting at each vertex, respectively ($3 \leq p, q \leq 100$).

The input will be terminated with a test case with $p = q = 0$, which should not be processed.

The total number of test cases will not exceed 1000.

Output

For each test case, write a single line with three integers on it — the number of vertices, edges and faces of the respective regular convex polyhedron. If there is no such polyhedron, write three -1 s instead.

Example

standard input	standard output
4 3 6 4 0 0	8 12 6 -1 -1 -1

Problem S. Sequence

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

For given n , Vasya should evaluate n -th element of Wilson sequence:

$$w_n = (n - 1)! \bmod n.$$

Input

Input consists of one or more test cases. There are at most 10 000 cases in the input.

Each case consist of single line with an integer n ($1 \leq n \leq 1\,000\,000$). Input will be terminated with a test case where $n = 0$ which should not be processed.

Output

For each test case, write a single line with the answer. Adhere to the sample output format below as close as possible.

Example

standard input
1
2
3
5
6
21
0
standard output
Case #1: 1st Wilson number is equal to 0.
Case #2: 2nd Wilson number is equal to 1.
Case #3: 3rd Wilson number is equal to 2.
Case #4: 5th Wilson number is equal to 4.
Case #5: 6th Wilson number is equal to 0.
Case #6: 21st Wilson number is equal to 0.