Problem A. Lottery

Input file:	stdin
Output file:	stdout
Time limit:	1 second
Memory limit:	256 megabytes

In recent years, televised lotteries have been rapidly losing ratings, ceding a large portion of the market to Internet casinos and bookmaker agencies. In an attempt to correct this trend and increase viewers' loyalty, the organizers of one of the lotteries has introduced a new fair and transparent method for determining the winner.

All lottery tickets are numbered sequentially with numbers from L to R (inclusively). Based on the original ticket numbers and a certain random number X, a sequence of encoded numbers A is generated, of which the i-th element is equal to $(L + i - 1) \oplus X$ $(1 \le i \le R - L + 1)$, where $\oplus$ refers to a bitwise addition of binary numerals using the "exclusive or"operation.

The show invites a special guest star, who does not know the value of numeral X, to choose some number K $(1 \le K \le R - L + 1)$. After this, the winning ticket is announced as the one whose encoded number is the K-th element of the sequence A in ascending order.

Input

The only line contains the numbers L, R, X and K $(0 \le L \le R \le 10^{18}, 0 \le X \le 10^{18}, 1 \le K \le R-L+1)$.

Output

The original number of the winning ticket.

Examples

stdin	stdout
6 15 0 7	12
1 20 10 5	14

Problem B. Street Workout

Input file:	stdin
Output file:	stdout
Time limit:	1 second
Memory limit:	256 megabytes

Street Workout is a fairly new term for athletes who perform strength exercises and tricks on everyday pullup bars in neighborhood courtyards or parks. Of course, many young people have trained on pullup bars and performed difficult elements in the past, but with the arise of social networks and video hosting, Street Workout has reached a global scale.

So for those of us who can only do two or three pullups at a time, how do you get started? The answer is simple and nothing new - start with "Stairs". This is a group game where the participants take turns approaching the bar, doing the same number of pullups. The number of pullups increases with each new approach until it reaches a certain maximum value, and then it begins to decrease. The game is finished when the number of pullups per turn reaches zero.

To make the training truly effective, it is important to choose the correct maximum number of pullups to do per turn in the game. Let's look at one variation of the game that allows two participants to train together, even if their skill levels are unequally matched.

Let's assume the first player's skill level is equal to A, while the second player's skill level is equal to B. We will designate the number of pullups per turn for the first player with the number N, and for the second player with the number M. At the first approach, as a mandatory warm-up, both players do one pullup each (i.e. initially $N = M = 1$). After both players have completed the first approach, the following rule will apply: if $N \times A < M \times B$, then the number N increases by one and the first player takes a turn, but if $N \times A > B \times M$, then the number M increases by one and the second player takes a turn. The exception is when one of the players is supposed to take two turns in a row (not counting the very first approach), or when a turn has been completed and the condition $N \times A = B \times M$ is met; in these situations, the rule described above does not apply, and each player repeats each of his turns, performing them in reverse order, until the game ends.

Your job is, knowing the skill level of the players, to find out the total number of pullups the two players will complete during the game.

Input

The first line contains two integers A and B ($1 \le A, B \le 10^9$) - the skill levels of the first and second players, respectively.

Output

The total number of pullups performed by the two players during the game.

Examples

stdin	stdout
3 5	18
1 2	8

Problem C. Routes in Disorder

Input file:	stdin
Output file:	stdout
Time limit:	1 second
Memory limit:	256 megabytes

In the 90's, a huge number of shuttles suddenly appeared on the streets of Russian cities (these are small buses or vans that run frequently along a route but stop wherever passengers ask). These were not only a faster and more convenient type of transportation, but also introduced a sense of chaos to the public transportation system. Many drivers regularly strayed from their designated routes, trying to detour around traffic jams and pick up more passengers, while some of these shuttles traveled around the city without any markings at all. The city administration was not happy with the situation, so it was decided to tighten control on following designated routes and set up a new numbering system for the shuttle.

There are N intersections in the city, numbered from 1 to N, as well as M two-way roads that connect certain pairs of intersections. There is at least one path between each pair of intersections.

It was decided to set up R routes. Assume that a particular route has a sequence of reference intersections $A_1, A_2, \ldots, A_k$. A shuttle should start its route from A_1,then follow the shortest path (by number of intersections crossed) to A_2, from A_2 the same way to A_3, from A_i to A_{i+1} (for $1 \le i \le k-1$), and finish its route at the intersection A_k. In addition, if a pair of intersections A_i and A_{i+1} has more than one shortest route available, the lexicographically smaller one is chosen.

Let's look at two different paths X and Y with an identical number of points k. Path X is lexicographically smaller than path Y, and if i $(1 \le i \le k)$ exists, then $X_i < Y_i$, and for all of j $(1 \le j < i)$ it is true that $X_j = Y_j$.

If we take all the intersections that shuttles cross on the path from A_1 to A_k and write them down in order B_1, B_2, ..., B_s, the route number will be the sum $\sum_{i=1}^{s} iB_i$. If some of the routes end up with the same number, we add a special code to all the numbers except the first one. For example, if we have 4 routes with the number 8172, the first one will keep the number 8172, the second will be 8172#2, the third will be 8172#3, and the fourth will be 8172#4.

Your task is to compute the numbers of all the routes using the plan described above.

Input

The first line contains two numbers N $(2 \le N \le 10^3)$ and M $(1 \le M \le 10^4)$.

The next M lines contain information about roads. Each road is described in a separate line using two numbers u and v $(1 \le u, v \le N, u \ne v)$. There is no more than one road between each pair of intersections. All the roads are two-way streets.

The next line contains the number R $(1 \le R \le 10^5)$ — the number of routes.

Next, the routes are defined in R lines, one per line. The description of the i-th route contains the number k_i $(2 \le k_i \le N)$ - the number of reference intersections on the i-th route and k_i numbers $A_{i,j}$ $(1 \le A_{i,j} \le N)$ — the numbers of the reference intersections on the i-th route in the order they are crossed. The reference intersections of a single route are all different.

The sum of k_i does not exceed $2 \cdot 10^5$.

Output

Output R lines, one per route. In the i-th line, print the number of the i-th route.

Example

stdin	stdout
13 17 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 5 5 7 1 11 11 12 12 2 11 13 13 10 10 9 9 4 11 10 4 5 1 3 13 12 7 8 1 2 3 11 13 12 4 7 3 1 7 2 2 6 8	644 644#2 211 40

Problem D. Security

Input file:	stdin
Output file:	stdout
Time limit:	1 second
Memory limit:	256 megabytes

Working as front-door security in the student dormitory is nerve-wracking for anyone. All day long, students rush past security, taking offense when they are asked to show ID, and worst of all, constantly letting the door slam!

One of the security officers got tired of losing his temper in vain, and decided to try to contain his frustration. He promised himself that over the course of the day he would not react to the students' behavior until the cumulative volume of the door slamming S either exceeded or equaled the limit to his patience L. As soon as this happened, the security officer would catch up with the student who slammed the door and give him a long lecture. After this, the security officer would calm down again, and the cumulative volume S would be nullified.

With a known volume of A_i for each time the door slams per day, you must calculate the number of lectures given by the security officer.

Input

The first line contains two numbers N ($1 \le N \le 1000$) and L ($1 \le L \le 1000$) — the number of times the door slammed and the limit to the security officer's patience. The second line contains N of the numbers A_i ($1 \le A_i \le 1000$) — the volume of the door slams.

Output

Number of lectures given per day.

Examples

stdin	stdout
6 3 1 3 2 2 3 2	3

Problem E. Numismatists

Input file:	stdin
Output file:	stdout
Time limit:	1 second
Memory limit:	256 megabytes

Numismatists are people who collect coins. Many of them do this as a hobby, trying to catch the spirit of an age gone by, while for others the coins are primarily a way to make a profit.

Most numismatists collect coins belonging to a specific epoch. To make it easier for buyers to find the coins they want, sellers are forced to keep the coins in correct order in the display case, sorting them by ascending order of minting date. The coins often get out of order while they are being examined and purchased, so to solve this problem a special coin-sorting algorithm was developed.

Let N be the number of coins in the display case. In a single operation, the seller can select a group of coins placed in a row in the display case, whose length must be equal to a power of two $(2, 4, 8, \dots)$, and move the oldest coin in the group to the beginning by taking all the coins in the group that were ahead of it and shifting them one position away from the beginning.

For a given number of coins, you must calculate the minimal number of operations necessary to put them in order by ascending minting date according to the algorithm described above, for the worst case scenario.

Input

The first line contains the number of coins in the display case N $(1 \le N \le 10^9)$.

Output

The necessary number of operations.

Example

stdin	stdout
3	3

Problem F. Tram Fans

Input file:	stdin
Output file:	stdout
Time limit:	2 seconds
Memory limit:	256 megabytes

Tram fans is a nickname for people who don't just believe that trams are a green form of transportation that reduce traffic, but who see trams as a true object of adoration, as well as the main attraction of any city. These are the people who organize committees and demonstrations to save the trams in their city, which local officials are trying to shut down, citing their supposed unprofitability. Tram fans spend their vacation traveling around the country or the world in hopes of seeing rare makes of trams and capturing them in photos.

A group of tram fans has arrived in a city that is rumored to have several rare makes of trams. Having procured a map of tram routes, they separate and set out on their search. When one of the tram fans sees a rare tram, he starts to call all his friends, and tells them the final destination of its route.

The city's tramway network is a connected set of N stops and $N - 1$ sections of rails between them. Each tram route is the shortest path between two particular stops. Your task, knowing the origin and destination stops A and B of a rare tram, is to determine the minimal number of stops that a tram fan who is located at stop C must travel in order to get onto the tram's route.

Input

The first line contains the number of stops in the tramway network N ($1 \le N \le 10^5$). The following $N - 1$ lines define number pairs U_i and V_i ($1 \le U_i, V_i \le N$) — the numbers of the stops that connect the i-th section of rails.

The next line defines the number of queries M ($1 \le M \le 10^5$). The following M lines contain query descriptions as three numbers A_i, B_i and C_i ($1 \le A_i, B_i, C_i \le N$), where A_i and B_i are the origin and destination stops of a rare tram, and C_i is the stop where a tram fan is located.

Output

Output M lines containing query responses, one per line. The response to the i-th query is the minimal number of stops on the journey of the tram fan from stop C_i to the route of the tram with origin and destination stops A_i and B_i.

Example

stdin	stdout
6 1 2 6 2 3 1 3 4 3 5 3 1 2 3 6 4 5 4 3 2	1 1 2

Problem G. Entrance Control

Input file:	stdin
Output file:	stdout
Time limit:	1 second
Memory limit:	256 megabytes

At any major event, whether it be a concert or a soccer game, turnstiles are installed at the entrance to the arena for safety reasons. People line up waiting to pass through the turnstiles, and police officers maintain order inside as the stadium fills up.

At a stadium entrance, N turnstiles are installed in a row. At certain intervals, several people are allowed to enter the stadium from a group of turnstiles that are next to each other (one person per turnstile). A group of K turnstiles is considered to be optimally controllable, if a number $p \geq 1$ exists for the group, where for each of the first p turnstiles pairs of fans have different grandstand section numbers on their tickets ($A_i \neq A_j$ for all $i \neq j, 1 \leq i, j \leq p$), and for everyone else ($i > p$) is true: $A_i = A_{i-p}$.

For each turnstile, we know the section number on the ticket of the first fan in line for this turnstile. You must learn how to determine whether a certain group of turnstiles in a row is optimally controllable at the given moment or not.

Input

The first line contains the number of turnstiles N ($1 \leq N \leq 10^5$). The second line contains N integers A_i ($1 \leq A_i \leq 10^5$) — the section numbers on the fans' tickets, one for each turnstile.

The third line contains the number of queries M ($1 \leq M \leq 10^5$). The following M lines contain query definitions, one per line. A query is defined by two numbers L_i and R_i ($1 \leq L_i \leq R_i \leq N$) — the code numbers of the turnstiles at the two ends, defining which turnstiles are included in the i-th group being checked.

Output

Print a line S of M symbols. If the group of turnstiles defined in query i is optimally controllable, then the i-th symbol in line S should be equal to 1, else 0.

Examples

stdin	stdout
10 7 1 2 3 1 2 3 3 1 7 7 1 10 1 4 1 5 2 6 2 7 2 8 8 10	0101101
5 1 1 1 2 1 4 1 3 1 4 1 5 3 4	1001

Problem H. Rehearsal Game

Input file:	stdin
Output file:	stdout
Time limit:	2 seconds
Memory limit:	256 megabytes

During a garage band rehearsal, the guitarist's strings kept going out of tune. Two other members of the band, the bassist and the drummer, randomly found a knight figure from a chess set in the garage and made up a new game to somehow pass the time.

The game is played on an endless chess board, plus the coordinates of the squares can be negative as well as positive. Before the game starts, the knight is placed on a square with the coordinates $(X_0; Y_0)$. The two players take turns moving. For each turn, the player must move the knight to one of 8 squares following the normal rules of chess, but on the i-th turn he must meet the condition $(|X_i| + |Y_i| < |X_{i-1}| + |Y_{i-1}|)$. If a player can't make a move, he loses.

Your task is to determine the winner and the total number of moves under the condition that both participants play optimally. In addition, we know that the winner tries to extend the game as long as possible, while the loser, on the other hand, wants to finish as quickly as possible.

Input

The first line contains the coordinates of the knight figure before the game starts X_0 and Y_0 $(|X_0| + |Y_0| \le 10000)$.

Output

In the first line, print «WIN» if the first player wins, or «LOSE» if the second player wins. Brackets or quotes do not need to be printed.

In the second line, print the total number of moves made by both players.

Examples

stdin	stdout
5 5	WIN 3
-8 3	LOSE 4

Problem I. Sponsored Testing

Input file:	stdin
Output file:	stdout
Time limit:	1 second
Memory limit:	256 megabytes

Everyone knows that many of the supposedly independent testing and reviews of audio equipment in specialized magazines are actually sponsored articles. In this problem, we will examine a rating approach that allows journalists to give the impression of objective testing while at the same time balancing the total rating points of all the tested equipment, thus staying on the good side of the manufacturers.

We are given a matrix of $N \times M$ numbers: the i-th row of the matrix represents the initial rating of the equipment numbered i, for each of M criteria. In a single operation, the journalist can subtract 1 from a cell in the matrix, while adding 1 to a different cell. We must calculate the minimum number of operations K that must be performed so that the total for each row of the matrix is the same, and the total for each column of the matrix is also the same. We know for sure that we will not need more than 10^5 operations to accomplish this.

Input

The first line contains the dimensions of the matrix, N and M ($1 \le N, M$; $N \times M \le 10^5$). Each of the following N lines contains M numbers — the i-th line of the matrix A ($0 \le A_{i,j}$). The sum of all the numbers in matrix A does not exceed 10^9.

Output

In the first line, output one integer — the number of operations K. In each of the following K lines, print 4 integers on each — the coordinates of the cell $(X_{i,1}; Y_{i,1})$ to subtract 1 from, and the coordinates of the cell $(X_{i,2}; Y_{i,2})$ to add 1 to. The coordinates must satisfy the conditions $1 \le X_{i,1..2} \le N$ and $1 \le Y_{i,1..2} \le M$. Note that after those operations matrix may contain several negative elements.

Examples

stdin	stdout
2 3 1 6 5 7 2 3	0
4 4 1 1 2 2 1 0 1 1 0 0 1 1 1 1 1 2	3 1 3 2 1 1 4 3 2 4 4 3 2

Problem J. Effect Pedals

Input file:	stdin
Output file:	stdout
Time limit:	1 second
Memory limit:	256 megabytes

Certain guitar players are literally crazy about using various effect pedals. Using a combination of multiple pedals can allow a guitarist to create unique sounds.

One guitarist who spent an entire day experimenting with different combinations of effect pedals finally achieved the sound of his dreams. Naturally, he immediately told his friends of his success and invited them to come over and listen. However, while he was on the phone, his younger brother decided to play his guitar...

Noticing multiple pedals connected sequentially, some of which were turned off and passed the sound on without any alteration, the little brother turned them all on as an experiment. But before he could even touch the strings, his furious older brother came running in and accused him of ruining his pedal configuration.

Wanting to diffuse the situation, the younger brother assured the elder that he didn't change the order of the pedals on the board, but only turned them all on. Besides, he remembered that before he touched anything, a certain nonzero number of pedals were turned on, and no two successive pedals were turned on.

To convince his brother that it would be easy to restore the original pedal configuration, the younger brother must calculate how many different pedal configurations exist that satisfy the conditions he described.

Each configuration is a sub-sequence of pedals that are in the "on"state. Two configurations are considered identical if the pedals in them match in each position. Pedals that are turned off do not affect the sound quality and are not taken into account when comparing configurations.

Input

The first line contains a sequence for turning on effect pedals S, consisting of lowercase Latin letters ($1 \le |S| \le 10^5$). Effect pedals that are denoted using the same letter and located in different positions are considered identical.

Output

The unknown number of various configurations for turning the pedals on, taken modulo $10^9 + 7$.

Examples

stdin	stdout
abcc	5

Note

In the first test example, the following pedal configurations are suitable: a, b, c, ac, bc.

Problem K. Random Order

Input file:	stdin
Output file:	stdout
Time limit:	1 second
Memory limit:	256 megabytes

Audiophiles usually prefer to listen to albums in their entirety, without changing the order of the tracks. It is generally believed that any disruption of the order that the performer intended inhibits the musical experience.

But today an audiophile was cruelly disappointed. In a book about his favorite rock group, he read that the order of the songs on one of their albums was changed by the producer before releasing it, and without getting the musicians' approval.

This incident upset the audiophile so much that he decided to listen to the tracks on this album only in random order. He intends to use the shuffle mode on his CD player.

The shuffle feature works according to the following rules:

1) The position of each track after shuffling must be different than the position it was in before shuffling.

2) If two tracks were next to each other before shuffling, they must not be next to each other after shuffling.

For a given number of songs N, your task is to output any order of tracks on the album that satisfies the shuffling rules, if all the tracks are initially numbered from 1 to N.

Input

The number of tracks on the album N ($1 \le N \le 10^5$).

Output

In N lines, output the numbers of the tracks after shuffling, one per line, following the conditions. If more than one answer is possible, print any one of them. If there is no possible order of tracks that would satisfy the shuffling rules, print «No solution» (without quotation marks) on a single line.

Examples

stdin	stdout
2	No solution
4	2 4 1 3