Problem A. Adventure in Flatland

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Flatland is a two-dimensional plane. Points with both coordinates different from zero are called *free*. Points with at least one zero coordinate are called *custom* points; when passing through these points one should pay 1 flatland dollar as a fee.

Adventurer stands at the free point with integer non-zero coordinates x_1 and y_1 and the goal of his adventure is to reach the free point with non-zero coordinates x_2 and y_2. He is allowed to choose any route he wants. Calculate minimum possible fee to be paid by adventurer.

Input

Input contains four integers x_1, y_1, x_2 and y_2 — coordinates of the start point and the finish point respectively ($x_1 \neq 0$, $y_1 \neq 0$, $x_2 \neq 0$, $y_2 \neq 0$, $-10\,000 \leq x_1, y_1, x_2, y_2 \leq 10\,000$).

Output

Print minimum possible fee in flatland dollars to be paid by adventurer.

Examples

standard input	standard output
25 11 -20 -20	1
20 20 20 21	0

Problem B. Basketball Plus-Minus

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

In a modern professional sport it is common for team managers to make game-related decisions based on statistics instead of personal impression. That is even more common for games that feature repetitive actions and the same player's dispositions many times per game, and each team is expected to score tens of times. Basketball is such a game and this task is about popular plus-minus statistics for individual players.

Each team has ten players for a game. At every moment of time each team has exactly five players on the court and five players in reserve. For the purpose of this problem we consider that substitutions are unlimited and can take place at any moment of time. Moreover, there are no restrictions on the number of substitutions for each particular player.

At the beginning of the game the individual score of each player is 0. Every time some team scores x points ($x = 1$, $x = 2$ and $x = 3$ are the options), the individual score of each player of this team who is currently on the court is increased by x. For the opposing team, the individual score of each player who is on the court is reduced by x. Individual score can become negative. No score changes take place for the players of both teams who are currently in reserve.

You are given the protocol of the game that has information about the players on the court at the beginning of the game, substitutions and scoring events.

Your task is to compute the individual scores of the plus-minus statistics for all players that appeared on the court at least once.

Input

The first line of the input contains the name of the first team. Then follow five lines containing names of the players, who are playing for the first team at the beginning of the game.

Then follow six lines containing the name of the second team and five names of the players of the second team that are on court at the beginning of the game.

Then goes the description of the game protocol. The first line of this description contains a single integer q ($1 \le q \le 1000$) — the number of events to consider.

Then follow q events listed in chronological order.

Each event is of one of two types.

- Events of the first type have form "`Team x scored n`", where x is one of two team names and n is an integer between 1 and 3.
- Events of the second type have form "`Team x replaced y with z`", where x is one of two team names, y is the name of the player from team x who is guaranteed to be on the court at that moment of time and z is the name of the player from team x who is guaranteed to be at reserve at that moment of time.

All names are non-empty strings of no more than 80 characters. Lowercase and uppercase English letters are allowed. Team's names are distinct, there are no two players with the same name in one team.

Output

For each player that appeared in the play (on the court) at least once print his name, team and individual plus-minus score on a separate line.

List the players in the order they are mentioned in the input for the first time. First, print the name of this player, then print the name of this player's team in brackets "`()`" (see sample output), finally print

the score. Positive scores should be printed with '+' sign and negative with '-'. Zero scores should have no sign at all.

Example

standard input
Lakers James Davis Howard Green ColdwellPope Clippers Leonard George Zubac Beverley Morris 10 Team Lakers scored 2 Team Clippers replaced George with Williams Team Clippers scored 3 Team Clippers scored 1 Team Lakers replaced Howard with Caruso Team Lakers replaced Green with Morris Team Lakers scored 2 Team Clippers replaced Zubac with George Team Lakers scored 2 Team Lakers scored 3
standard output
James (Lakers) +5 Davis (Lakers) +5 Howard (Lakers) -2 Green (Lakers) -2 ColdwellPope (Lakers) +5 Leonard (Clippers) -5 George (Clippers) -7 Zubac (Clippers) 0 Beverley (Clippers) -5 Morris (Clippers) -5 Williams (Clippers) -3 Caruso (Lakers) +7 Morris (Lakers) +7

Problem C. C and Pascal Strings

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

In this problem we will deal with byte sequences. Each byte is given by two hexadecimal digits (four bits each, eight bits in total) and is treated as an unsigned integer between `0` and $\texttt{ff}_{16} = 255_{10}$ (inclusive). Hexadecimal digits are denoted `0123456789abcdef`, as usual.

Consider the C ASCII string to be the byte sequence starting with zero or more bytes from range between $\texttt{20}_{16}$ and $\texttt{7f}_{16}$ (inclusive) followed by a zero byte. There may be arbitrary bytes following the zero byte, they are called "junk" bytes.

Consider the Pascal ASCII string to be the byte sequence starting with byte l defining the length of the string, followed by l bytes from range between $\texttt{20}_{16}$ and $\texttt{7f}_{16}$ (inclusive). There may be arbitrary bytes following the last of these l bytes, they are also called "junk" bytes. In particular, Pascal string length is limited to the range from `0` to $\texttt{ff}_{16} = 255_{10}$.

You are given a memory dump as a sequence of integers b_i given in hexadecimal form. Each integer belongs to the range from `0` to $\texttt{ff}_{16}$ (inclusive). Your task is to determine, which of four cases takes place:

- dump defines a valid C ASCII string as well as a valid Pascal ASCII string;
- dump defines a valid C ASCII string, but not a valid Pascal ASCII string;
- dump defines a valid Pascal ASCII string, but not a valid C ASCII string;
- dump does not define a valid C ASCII string or a valid Pascal ASCII string.

Note that in all cases junk bytes are allowed.

Input

Input contains a sequence of no more than 1000 space-separated hexadecimal integers; each integer consists of exactly two characters, each of which is from one of the `0123456789abcdef` characters. In particular, leading zeroes are allowed.

Output

If the dump can be interpreted both as a C ASCII string and as a Pascal ASCII string, print the word "`Both`".

If the dump can be interpreted as a C ASCII string but not as a Pascal ASCII string, print the word "`C`".

If the dump can be interpreted as a Pascal ASCII string but not as a C ASCII string, print the word "`Pascal`".

If the dump can't be interpreted as a C ASCII string or as a Pascal ASCII string, print "`None`".

Examples

standard input	standard output
4d 4f 53 43 4f 57 00 5a	C
04 49 43 50 43 00	Pascal
05 4e 4f 4e 45 81	None
00 f4	Both

Problem D. Deep Primes

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

For any non-negative integer x we can consider its decimal representation as a string of digits from 0 to 9. Denote this string as $S(x)$. On the opposite, for any string s of decimal digits we can get an integer it represents by neglecting all leading zeroes, except maybe one to represent integer 0. Denote this integer $D(s)$.

We say that integer y is an *integer-substring* of integer x if there exists a string s that is a substring (consecutive subsequence) of $S(x)$ and $D(s) = y$.

Recall that integer x is called prime if it is positive and has exactly two divisors. First five primes are $2, 3, 5, 7, 11$.

We call integer x *deep prime* if it is prime and any y that is integer-substring of x is prime.

You are given two integers n and m, calculate the number of deep prime integers between n and m, inclusive.

Input

The only line of the input contains two integers n and m ($1 \le n \le m \le 10^{18}$).

Output

Print one integer — the number of deep prime integers x that satisfy $n \le x \le m$.

Example

standard input	standard output
1 11	4

Problem E. Easy Money

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

In the June of 2345 Bomboslav went to planet Nibiru for a summer internship as a barista in Bakefooc Inc. office there. As Bakefooc Inc. is a US-based company they continue to pay interns with cheques. Pay cheque is a paper document issued by an employer to pay an employee. Typically it is a piece of stamped paper with a sum and a signature of a head of payment department on it. You just bring this to any bank branch, wait for several days and enjoy the money being transferred to your account — what an easy and convenient way to be paid!

After two or three payments an evil plan came to Bomboslav's mind. What if he can scrap the upper layer of paper from the part where digits are written and stick them back in some order he might prefer more?

So he received his next pay cheque for n nibidollars and successfully scrapped all the digits from it. He now wants to put them back in a following way.

- All digits must be placed back to form a single valid integer (no leading zeroes are allowed) so no scrapped spot is visible.
- Bomboslav believes in magic of numbers and wants to increase the chances of his shady transaction by obtaining an integer that is divisible by 7.
- The resulting integer should be as large as possible (Bomboslav is a greedy intern).

What is the best result he can get? Note that he is not allowed to form the original integer n if it was not divisible by 7.

Input

The only line of the input contains a single integer n ($1 \le n < 10^{1000}$) — the original sum in the pay cheque.

Output

If there is a way to obtain at least one integer that satisfies all the requirements, print maximum such integer in the only line of the output. Otherwise, print `-1`.

Examples

standard input	standard output
14	14
11	-1
2020	2002

Problem F. Fun at Luggage Claim

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

In most cases it's not about whether your job is fun or not, it's about whether you know how to have fun or not. This task is about Olya and Kolya — staff members of Vnumometyevo airport who certainly do have fun while working at a luggage claim.

Luggage belt consists of n consecutive cells numbered from 1 to n. As it is a belt, cells numbered i and $i + 1$ are adjacent for any i from 1 to $n - 1$ as well as cells n and 1.

At the beginning of the Luggage Belt Game Olya puts a_i items of luggage at the i-th cell of the belt. She also tells Kolya a sequence $b_1, b_2, \ldots, b_n$ that describes a desired distribution of the luggage among the belt cells. While all passengers are stuck at the arrival passport control Kolya has enough time to apply an arbitrary number of operation of the following type. He can pick any cell i such that it has **at least two** items of luggage more than each of two adjacent cells, and move one item of luggage from cell i to each of them.

Is it possible for Kolya to achieve a state where the i-th belt cell has exactly b_i items of luggage in it?

Input

The first line of the input contains a single integer n ($3 \le n \le 100\,000$) — the length of the luggage belt.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$).

The input ends with the third line containing n integers $b_1, b_2, \ldots, b_n$ ($0 \le b_i \le 10^9$).

Output

If it is possible for Kolya to achieve luggage distribution provided by Olya print "Yes" in the only line of the output. Otherwise, print "No".

Examples

standard input	standard output
3 0 0 2 1 1 0	Yes
3 0 2 0 0 1 1	No
4 0 100 0 10 33 40 33 4	Yes

Problem G. Game With Stones

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Porfiry and Rodion are playing with stones. They have a pile of m stones, and n boxes, arranged in a row. They consider different distributions of all stones to boxes. Each such distribution can be described with n non-negative integers $a_1, a_2, \ldots, a_n$ such that $a_1 + a_2 + \cdots + a_n = m$.

Rodion loves to move stones from a pile to adjacent one. In one step, he can take exactly one stone from some non-empty box and put it to one of adjacent boxes. For two distributions $a = (a_i)$ and $b = (b_i)$, Porfiry defined value $W(a, b)$ as minimum number of Rodion's moves required to transform distribution a to distribution b.

After some observation, Porfiry gave Rodion the following task: given k distributions $(a_i^1), (a_i^2), \ldots, (a_i^k)$, find distribution b, such that the total distance $W(a^1, b) + W(a^2, b) + \cdots + W(a^k, b)$ is minimum possible.

Input

The first line of the input contains three space-separated integers n, m and k ($1 \le n, k \le 1000$, $1 \le m \le 10^9$).

The j-th of the following k lines contains a description of the j-th distribution, represented by n non-negative integers $a_1^j, a_2^j \ldots, a_n^j$ ($a_1^j + a_2^j + \cdots + a_n^j = m$).

Output

Output optimal distribution b. If there is more than one optimal answer, print any of them.

Example

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

Problem H. Hungry Cannibals

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

A group consisting of a cannibals and b missionaries are travelling together. They are facing a river with a single boat docked at their side of the river. Boat is pretty small; its capacity is enough for at most two people simultaneously. Moreover, boat is not equipped with any kind of motor. The only way to move the boat is to use a paddle, which requires some skill and physical strength.

It turned out that only x among a cannibals and only y among b missionaries are able to use the paddle. In order to sail the boat from one side of the river to the other, at least one of the passengers must be able to use the paddle.

Cannibals would not be cannibals if they were not seeking for human flesh. Namely, if there appear to be p cannibals and q missionaries at the same side of the river (possibly including the docked boat) such that $p > q$ and $q > 0$, missionaries become the part of cannibal dinner which is not acceptable for the purposes of their journey. In other cases (when $p \le q$ or $q = 0$) cannibals behave like civilized people and do follow the accepted transfer plan.

Taking into account all these restriction, find out if there exists a way to transfer all people to the other side of the river or not.

Input

The first line of the input contains an integer T ($1 \le T \le 1000$) denoting the number of test cases.

Each of the following lines contains four integers a, x, b, y ($0 \le a \le b \le 1000$, $0 \le x \le a$, $0 \le y \le b$, $a + b > 0$).

Output

For each test case print `Yes` or `No` depending on whether it is possible for all people to have a safe travel to the other side of the river or not.

Example

standard input	standard output
4 2 0 2 1 1 1 4 0 2 1 2 0 3 1 3 1	Yes Yes No Yes

Note

In the first test case one of the possible ways to transfer everybody to the other side of the river is the following.

- The missionary able to paddle will occupy one of the places in the boat.
- He starts by transferring one cannibal to the other side. Original side of the river contains one missionary and cannibal, and same for the boat, so nobody is eaten.
- He leaves cannibal at the other side of the river and returns back to the original side.
- It contains 2 missionaries and 1 cannibal now, so everybody is still safe.

- He picks one missionary and sails to the other said of the river, leaving missionary there.
- He returns back, picks the remaining cannibal and finally sails to the other side of the river.

In the second case there is only one cannibal, so he cannot eat anybody under any circumstances. So, any way of transferring everybody to the other side of the river is valid.

In the third test case it is not possible to avoid the situation when two cannibals and one missionary are located on the same side of the river.

Problem I. Integer Number Format

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

A new format IIII-4 of storing integer numbers is being developed. Encoding of each integer results in one, two, three, four or five 4-bit groups. First group is mandatory and then follow from zero to four optional additional groups.

Decoding is done as follows.

1. Read first 4-bit mandatory group.
2. Look it up in a table that contains the number of additional groups g and the offset f.
3. Read g additional groups. Decode this groups as an integer y from 0 to $2^{4g} - 1$.
4. Add offset to y, i.e. the result is $y + f$.

You are given two integers x and y and a sequence of integers $a_1, a_2, \ldots, a_n$, such that $x \le a_i \le y$. Your goal is to decide on a look up table and offsets in a way that allows to encode and decode all integers from x to y inclusive and the length of the encoding of the input sequence is minimum possible.

Note that the order of bits that is used to decode the number from 0 to $2^{4g} - 1$ doesn't matter for the purpose of this problem.

There should be a unique way to encode each integer from x to y using this table. However, integers outside of this range can be encoded in any number of ways.

Input

The first line contains two integers x and y ($-10^9 \le x \le y \le 10^9$, $y - x \le 1\,000\,000$) — bounds of the range that we need to be able to encode with the method.

The second line contains a single integer n ($1 \le n \le 100\,000$) — the length of the sequence to be encoded. The third line contains n integers $a_1, a_2, \ldots, a_n$ ($x \le a_i \le y$).

Output

Print the optimal table represented by 16 lines corresponding to decoding information of 16 prefixes.

Each line should contain two integers: g_i ($0 \le g_i \le 4$) stands for the number of additional groups and f_i ($-2 \cdot 10^9 \le f_i \le 2 \cdot 10^9$) is the offset.

The given sequence of numbers must be encoded to the minimum number of bits by the table. It is guaranteed that at least one valid encoding exists. If there are several optimal tables, you can print any of them.

Examples

standard input	standard output
0 15 16 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	0 0 0 1 0 2 0 3 0 4 0 5 0 6 0 7 0 8 0 9 0 10 0 11 0 12 0 13 0 14 0 15
-128 127 4 1 0 -1 0	2 -261 0 -5 0 -4 0 -3 0 -2 0 -1 0 0 0 1 1 2 1 18 1 34 1 50 1 66 1 82 1 98 1 114

Problem J. Just Different Rules...

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

There are multiple playing cards. Each card has two sides, black and white, and a rank, which is a number from 1 to n. There may be several cards of the same rank.

You're trying to solve a puzzle. In the puzzle all cards are arranged into m lanes. In one turn you can simultaneously flip all cards of a certain rank (that is, black cards become white and vice versa). If every lane contains at least one white card, you win.

Having struggled for several hours, you finally solved the puzzle... only to find out that you misinterpreted the rules: there should be *at least* one white card in every lane, and you got into a position where there is *at most* one. Disappointed, you made a series of irrelevant flips.

Can you solve the puzzle according to its original rules?

Input

In the first line of the input there are two integers n and m ($1 \le n, m \le 200\,000$), the number of ranks and lanes respectively. Next m lines describe lanes.

The description of each lane starts with integer k ($k \ge 1$), the number of cards in this lane. It is followed by k non-zero integers with absolute value at most n which describe cards. Positive integer x denotes a white card of rank x, while integer $-x$ denotes a black card of rank x.

The total number of cards in all lanes does not exceed 500 000.

It is guaranteed that there exists a sequence of flips leading to a position where each lane contains **at most** one white card. Note, however, that your program should find a position with **at least** one white card in each lane (it is not necessarily possible).

Output

If it is impossible to reach a position with at least one white card in each lane, print "`No`".

Otherwise, print "`Yes`" and integer f — the number of required flips. Then print f distinct integers between 1 and n, denoting the ranks to flip.

If there are multiple solutions, print any of them.

Examples

standard input	standard output
4 3 2 1 -2 2 -1 -3 3 1 3 4	Yes 1 3
1 2 1 1 1 -1	No

Note

In the first sample, one of the possible ways to reach the position with **at most** one white card in each lane is to flip cards of rank 1 and cards of rank 4. If you flip cards of rank 3, you reach the position with **at least** one white card in each lane.

Problem K. King and Zeroing

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

There are n cities in some imaginary kingdom. All cities are connected by $n - 1$ bidirectional roads. The toll to drive a single road is exactly 1 credit in each direction.

Recently citizens have started to wonder why it's so expensive to drive across the country. King has decided to make some roads free of charge. He wants to choose a single road for each city and make the toll in the **outgoing** direction free for everybody. His goal is to minimize the maximum total toll among all direct road trips from one city to another. You should help him!

Input

The first line contains a single positive integer n ($2 \le n \le 200,000$) — the number of cities in an imaginary kingdom.

Next $n - 1$ lines contain two positive integers a_i and b_i ($1 \le a_i, b_i \le n$) — the description of i-th road, connecting cities a_i and b_i.

Output

In the first line print a single integer d — the minimum total toll which would be required to pay for the most expensive road trip from one city to the other.

In the next line print n integers e_i — the index of the road selected for the i-th city to be free of charge. If no road is selected for the i-th city print $e_i = -1$.

Examples

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

Problem L. Linear Algebra Intensifies

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

The old linear algebra professor from a previous Moscow subregional problem keeps torturing his students with massive homework assignments. He came up with a way to describe a huge symmetric square matrix faster than dictating all its elements.

Professor chooses a positive integer n and names m ranges $[l_1, r_1], \ldots, [l_m, r_m]$ such that for all $i = 1, \ldots, m$ both l_i and r_i are integer, and $1 \le l_i \le r_i \le n$. The $n \times n$ matrix A is then constructed as follows: for any pair of indices x, y, the matrix element $A_{x,y}$ is equal to the number of indices i ($1 \le i \le m$) such that both x and y belong to the range $[l_i, r_i]$. For example, for $n = 3$ and the list of ranges $[1, 2], [2, 3], [1, 2], [3, 3]$, the resulting matrix A is $\begin{pmatrix} 2 & 2 & 0 \\ 2 & 3 & 1 \\ 0 & 1 & 2 \end{pmatrix}$.

Given n and the list of ranges, professor then asks students to compute determinant of the resulting matrix A. Professor doesn't care for comparing huge numbers, so he asks for the answer to be computed modulo his favourite prime number 998 244 353.

Professor may be cruel, but he is a reasonable teacher: he agrees there's little merit in exercise when writing down the problem is harder than actually solving it afterwards. Thus, in his assignments m is never much greater than n, more precisely, $m \le n + 300$.

After a few semesters in professor's class, you are very much done with his attitude and decided to write a program that will compute the answer for you.

Input

The first line contains two integers n and m ($1 \le n, m \le 500\,000$, $m \le n + 300$).

The following m lines describe the ranges named by professor. The i-th of these lines contains two integers l_i, r_i ($1 \le l_i \le r_i \le n$).

Output

Print a single integer — determinant of the matrix A modulo 998 244 353.

Example

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

Problem M. Matrix 4

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Matrix 4 is coming! There are rumors that one of the main plot twists of this ongoing epic movie will be Neo searching for Trinity in a 4-dimensional labyrinth.

Each point of the labyrinth may be seen as a matrix 2×2. Neo is standing at the matrix $S = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$.

Trinity is held captive at the matrix $F = \begin{pmatrix} p & q \\ r & s \end{pmatrix}$.

From each matrix of labyrinth you can go to one of 4 directions which are denoted by letters a, A, b, B. When taking direction $c \in \{a, A, b, B\}$ from the matrix X, you reach matrix XM_c (matrix product of X and M_c) with M_c defined as follows:

$$M_a = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix} \quad M_A = \begin{pmatrix} 1 & -2 \\ 0 & 1 \end{pmatrix} \quad M_b = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix} \quad M_B = \begin{pmatrix} 1 & 0 \\ -2 & 1 \end{pmatrix}$$

Your task is to help Neo go from S to F or to decide that it is impossible. If it is possible, you should also provide Neo with a description of the route which should be taken in order to reach F. Route should be expressed as a sequence of steps; steps may be *repetition groups* that look like a sequence of steps repeated arbitrary positive integer number of times; repetition groups may possibly include other repetition groups.

Formally, route is defined using the following grammar:

```
<route> := <empty> | <step><route>;
<step> := a | A | b | B | (<route>)<count>;
```

where `<empty>` is an empty string and `<count>` is a positive integer.

To follow the route means to follow each step of the route from left to right. To follow the step means either take corresponding direction or follow the inner route `<count>` times.

Input

The first line of the input contains T ($1 \le T \le 10\,000$), the number of test cases.

Each of the following T lines contains four integers p, q, r, s ($-1\,000\,000 \le p, q, r, s \le 1\,000\,000$) defining the components of the matrix $F = \begin{pmatrix} p & q \\ r & s \end{pmatrix}$ in which Trinity is held captive.

Output

For each test case either print word "`Impossible`" or print a route from S to F. Do not print any extra spaces, strictly follow the formal grammar above. Empty route is allowed.

Number of repetitions in any step that is a repetition group should not exceed 10^9.

Route length should not exceed one KiB (1024 bytes), i.e. the length of the line with route description should not exceed 1024.

Checking program will evaluate your route using fast matrix exponentiation. If at any moment any intermediate matrix has component with absolute value greater than 10^9, you get Wrong Answer verdict for the test.

Example

standard input	standard output
3 -7 4 -2 1 25 12 -48 -23 -1 0 0 1	aaB (B(a)3b)2 Impossible

Note

In the first case, route takes Neo through the following sequence of matrices:

$$\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix} \to \begin{pmatrix}1 & 2\\0 & 1\end{pmatrix} \to \begin{pmatrix}1 & 4\\0 & 1\end{pmatrix} \to \begin{pmatrix}-7 & 4\\-2 & 1\end{pmatrix}$$

Route from the second case is equivalent to the following route: `BaaabBaaab`.

Problem N. New Randomized Go

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

The game of go was thought to be among the last to see computer vs human competition becoming pointless because of humiliation of the former. Alas, AlphaGo was there to tear apart the hopes of the community and Lee Sedol is now looking for new ways to compete with someone at something. He came up with a randomized version of the game that asks for no thinking at all, only pure luck is required. Moreover, this new game is for one player only!

Consider the circle of integer perimeter l. Introduce a coordinate system along the circumference so that any its point is assigned a real value x, $0 \le x < l$. We are given n distinct points on the circumference, the i-th point has coordinate x_i.

The game process is very simple. The player throws a fair coin n times and paints each point red or blue depending on the outcome of the corresponding throw. Thus, all points are independently and equiprobably assigned one of these two colors. Then, the players draws a convex hull of all red points and a convex hull of all blue points. Recall that the convex hull of a finite set of points is the smallest convex polygon such that all points of the set lie inside the polygon or on its border.

The player is declared a winner if the center of the circle is a part (lies inside or on the border) of both convex hulls. You are given the parameters of the game before the coin part starts. Compute the probability of a winning game.

Input

The first line contains two integers n ($1 \le n \le 1\,000\,000$) and l ($n \le l \le 10^9$).

Then follow a line containing n integers $x_1, x_2, \ldots, x_n$ ($0 \le x_i < l$, $x_i \ne x_j$ for $i \ne j$) describing the positions of the points along the circumference.

Output

It is guaranteed that the probability of a winning game can be expressed as an irreducible fraction $\frac{p}{q}$, where q is coprime with $10^9 + 7$. Your goal is to find such integer r ($0 \le r < 10^9 + 7$) that $r \cdot q \equiv p \pmod{10^9 + 7}$.

Examples

standard input	standard output
4 100 0 30 50 80	125000001
8 100 1 12 34 45 51 84 88 92	515625004

Note

Probability is $\frac{1}{8}$ in the first test case and $\frac{25}{64}$ in the second test case.