

Problem A. Polynomial in a Black Box

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

This is an interactive problem.

Alice has a black box which works with integers modulo $m = 10^9 + 7$. If a user types a number x on the keyboard of the box, the screen shows the number equal to the value of the polynomial $p(x) = (a_d x^d + a_{d-1} x^{d-1} + \dots + a_1 x^1 + a_0) \bmod m$. The degree d of the polynomial is unknown, as are its coefficients a_i . It is only known that $0 \leq d \leq 10$ and $a_d \neq 0$.

Alice can type several numbers x and learn the values of the polynomial for these numbers. Help her find the degree d of the polynomial. She can input an x at most $d + 3$ times.

Interaction Protocol

To learn the value of the polynomial for number x , print a line of the form “**ask** x ” ($0 \leq x < 10^9 + 7$). As a result, you will get a line with the value $p(x)$, or, if you asked more than $d + 3$ such questions, you will get the number -1 instead of the value, and evaluation of your solution will terminate.

To give the answer, print a line of the form “**degree** d ”. After that, terminate your solution gracefully.

After printing each line, flush the output buffer, or you will get the outcome **Idleness Limit Exceeded**: this can be done by calling, for example, `fflush (stdout)` in C or C++, `System.out.flush ()` in Java, `flush (output)` in Pascal, or `sys.stdout.flush ()` in Python.

Example

standard input	standard output
1000000006	ask 1
7	ask 3
34	ask 6
98	ask 10
	degree 2

Note

In each test, the degree and the coefficients of the polynomial $p(x)$ are chosen and fixed in advance.

In the example, which is also the first test in the testing system, $p(x) = x^2 + 1\,000\,000\,005$. All other tests were created as follows: first, the degree d was chosen ($0 \leq d \leq 10$), and after that, one of the polynomials of such degree was chosen as $p(x)$ uniformly at random.

Problem B. Board Trick

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

Danila the Magician and Mia the Assistant are showing a trick with a board.

First, Danila secludes himself, while Mia asks the audience to fill the 8×8 square table drawn on the board: each cell can contain either zero or one.

Next, another member of the audience chooses the secret number: an integer from 1 to 64. The number is announced to Mia but not to Danila.

After that, Mia comes to the board and changes exactly one number on it: either a zero into one, or a one into zero.

Finally, Danila returns, looks at the board for a while, and happily announces the secret number he did not know.

How do they perform the trick? Help Danila and Mia to prepare and then show it.

Interaction Protocol

In this problem, your solution will be run twice on each test. Each test can contain several test cases.

During the first run, the solution acts as Mia the Assistant: given a board and a secret number, it has to change exactly one number on the board. The first line contains the word “**Mia**”. The second line contains an integer t : the number of test cases ($1 \leq t \leq 1000$). The test cases follow. Each consists of nine lines. The first eight lines contain eight space-separated integers each: the numbers on the board (each equal to either 0 or 1). The ninth line contains the secret number (an integer from 1 to 64).

To answer a test case, the solution must print nine lines. On each of the first eight lines, print eight space-separated numbers: the numbers on the board (each equal to either 0 or 1). When comparing to the input, exactly one of them must be different: either a zero must turn into one or a one must turn into zero. On the ninth line, print three “minus” signs.

During the second run, the solution acts as Danila the Magician: given the final state of the board, it has to guess the secret number. The first line contains the word “**Danila**”. The second line contains an integer t : the number of test cases, the same as during the first run ($1 \leq t \leq 1000$). The test cases follow. Each consists of nine lines. The first eight lines contain eight space-separated integers each: the numbers on the board, exactly the same as were printed (each equal to either 0 or 1). The ninth line contains three “minus” signs.

To answer a test case, the solution must print one line containing the secret number (an integer from 1 to 64).

During each run, each line of input including the last one is terminated by a newline.

Examples

On each test, the input during the second run depends on the solution’s output during the first run. In the example, we will consider a solution where Danila simply says the sum of all numbers on the board as a guess. And Mia tries to make this sum equal to the secret number. Surely, this solution does not always work.

The two runs of this solution on the first test are shown below.

standard input	standard outptut
Mia	1 0 0 0 0 0 0 0
2	0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0	---
0 0 0 0 0 0 0 0	0 0 1 0 1 0 0 1
1	1 0 0 1 0 1 1 0
0 1 1 0 1 0 0 1	1 0 0 1 0 1 1 0
1 0 0 1 0 1 1 0	0 1 1 0 1 0 0 1
1 0 0 1 0 1 1 0	0 1 1 0 1 0 0 1
0 1 1 0 1 0 0 1	1 0 0 1 0 1 1 0
0 1 1 0 1 0 0 1	1 0 0 1 0 1 1 0
1 0 0 1 0 1 1 0	0 1 1 0 1 0 0 1
1 0 0 1 0 1 1 0	---
0 1 1 0 1 0 0 1	
31	
Danila	1
2	31
1 0 0 0 0 0 0 0	
0 0 0 0 0 0 0 0	
0 0 0 0 0 0 0 0	
0 0 0 0 0 0 0 0	
0 0 0 0 0 0 0 0	
0 0 0 0 0 0 0 0	
0 0 0 0 0 0 0 0	
0 0 0 0 0 0 0 0	
0 0 0 0 0 0 0 0	

0 0 1 0 1 0 0 1	
1 0 0 1 0 1 1 0	
1 0 0 1 0 1 1 0	
0 1 1 0 1 0 0 1	
0 1 1 0 1 0 0 1	
1 0 0 1 0 1 1 0	
1 0 0 1 0 1 1 0	
0 1 1 0 1 0 0 1	

Problem C. Connectivity

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

This is an interactive problem.

The jury made a random undirected graph of n vertices and m edges with no loops or parallel edges. In each test, the graph is randomly uniformly sampled from all possible graphs with fixed n and m before the testing of your program starts.

Your program has to determine whether the graph is connected, while knowing only n and m at first. The program can perform a “? i ” query ($1 \leq i \leq n$) at most $2 \cdot n$ times. In response for such query, the testing system will give either:

- Positive integer j ($1 \leq j \leq n$) meaning that there is an edge between vertices i and j and that the edge was not previously communicated to the program (including any responses to “? j ” queries).
- Number -1 meaning that all edges adjacent to the vertex i were already communicated to the program.

Interaction Protocol

The first line of the input contains an integer T : the number of independent test cases for your program to handle. Your program will then have T interactions with the testing system.

An interaction starts with integers n and m , communicated to your program on a separate line: the number of vertices and edges in the secret graph ($2 \leq n \leq 10^4$, $1 \leq m \leq \min(\frac{n(n-1)}{2}, 10^4)$).

Afterwards, you can repeatedly print a request on a separate line in the following format: question mark without quotes (“?”, ASCII code 63), followed by a space, followed by an integer i ($1 \leq i \leq n$): the number of the vertex for which you want to know the next adjacent edge.

In response for each such query, the testing system will print either a positive number of another vertex adjacent to i , or -1 .

To give the answer, print a line with either “+” (plus sign, ASCII code 43) if the graph is connected, or “-” (minus sign, ASCII code 45) if the graph is not connected.

After printing the answer for the last T -th test case, terminate your solution gracefully.

The sum of n for all test cases in a run does not exceed 10^5 .

After printing each line, flush the output buffer, or you will get the outcome **Idleness Limit Exceeded**: this can be done by calling, for example, `fflush (stdout)` in C or C++, `System.out.flush ()` in Java, `flush (output)` in Pascal, or `sys.stdout.flush ()` in Python.

Example

standard input	standard output
2	
5 4	
	? 1
2	
	? 4
1	
	? 1
5	
	? 1
-1	
	? 4
3	
	? 4
-1	
	+
5 4	
	? 5
-1	
	-

Note

Empty lines are added for clarity, they are absent during interaction.

In the example test, the two following graphs are used. The testing system responds to “? i ” with the first edge in the list which is adjacent to i and was not yet communicated to the program.

1. $n = 5$, $m = 4$, edges are ordered as follows: 1–2, 1–4, 3–4, 1–5.
2. $n = 5$, $m = 4$, edges are ordered as follows: 1–4, 1–3, 1–2, 3–4.

Problem D. Absolute Pairwise Distance

Input file: *standard input*
Output file: *standard output*
Time limit: 5.5 seconds
Memory limit: 512 mebibytes

John Doe invented a nice way to measure distance between two arrays of different length. Let a_1, \dots, a_{l_1} be the first array and b_1, \dots, b_{l_2} be the second one. Then $d(a, b) = \sum_{i=1}^{l_1} \sum_{j=1}^{l_2} |a_i - b_j|$. Unfortunately, this distance function does not satisfy the triangle inequality, but John decided to conduct a few experiments anyway.

John has a large array a_1, \dots, a_n . For q instances of values (l_1, r_1, l_2, r_2) , he would like to know the values $d((a_{l_1}, a_{l_1+1}, \dots, a_{r_1}), (a_{l_2}, a_{l_2+1}, \dots, a_{r_2}))$. Help him find these values.

Input

The first line contains two integers n and q : the number of elements in the array and the number of queries ($1 \leq n, q \leq 10^5$). The second line contains n integers a_1, \dots, a_n : the elements of John's large array ($0 \leq a_i \leq 10^8$). The next q lines contain four integers each: l_1, r_1, l_2, r_2 , which are the parameters of the respective query ($1 \leq l_1 \leq r_1 \leq n, 1 \leq l_2 \leq r_2 \leq n$).

Output

For each query, print the value of $d((a_{l_1}, a_{l_1+1}, \dots, a_{r_1}), (a_{l_2}, a_{l_2+1}, \dots, a_{r_2}))$ on a separate line.

Example

standard input	standard output
5 5	1
1 2 3 4 5	3
1 1 2 2	6
1 1 2 3	4
1 1 2 4	40
1 2 2 3	
1 5 1 5	

Problem E. Multiplication and Division by 2

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

Consider the number x stored in `uint32` data type. We can multiply or divide it by 2 any number of times in any order. Can we obtain the number y after some sequence of operations?

When a is stored in an `uint32`, and we multiply it by 2, it transforms into $(a \cdot 2) \bmod 2^{32}$. For example, $(3 \cdot 2) \bmod 2^{32} = 6$, and $(2\,147\,483\,649 \cdot 2) \bmod 2^{32} = 2$.

When a is stored in an `uint32`, and we divide it by 2, it transforms into $\lfloor \frac{a}{2} \rfloor$. For example, $\lfloor \frac{6}{2} \rfloor = 3$, and $\lfloor \frac{3}{2} \rfloor = 1$.

Input

The first line contains an integer t , the number of test cases ($1 \leq t \leq 1000$). The next t lines describe test cases, one per line. Each test case is given by two integers x and y ($0 \leq x, y < 2^{32}$).

Output

For each test case, print a single word on a separate line: “Yes” if we can turn x into y using the allowed operations, or “No” otherwise.

Example

standard input	standard output
2	Yes
2147483649 1	No
9 13	

Explanation

In the first test case, we can multiply x by 2, and then divide the result by 2 to get y .

In the second test case, there is no way to turn x into y .

Problem F. Festive Baobab

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

Do you know where is the baobab in the Math&Mech facility? Well, if no, ask someone after the contest to show you...

Everyone knows that baobab is a tree. So, as any tree, it has trunk and branches. Branches grow from the trunk or from other branches.

For September 1, this baobab is usually decorated to make new students feel comfortable. For this purpose, there is a large chest with small colorful decorations deep in the building dungeons. Every decoration weighs exactly one gram.

Unfortunately, the baobab is very old, therefore it will be broken if overloaded with decorations. Specifically, for every branch, you know the maximum weight of decorations it can carry. If a branch, together with all branches growing from it (directly or via other branches), carries decorations such that their total weight exceeds this limit, this branch will break. This is unacceptable. But you can put several decorations on a single branch unless it causes some branch to break.

Some branches are near the entry and well shown, others are hidden in the depths of the baobab. Therefore, every decoration can bring more or less joy, depending on its position. The total joy of the baobab is the sum of joys of the branches where the decorations are located. If there are several decorations on a single branch, its joy is multiplied by their quantity.

What is the maximum possible total joy the baobab can have?

Input

The first line contains two integers n and t : the number of branches on the baobab and the number of decorations ($1 \leq n \leq 100\,000$, $1 \leq t \leq 10^9$). Each of the next n lines contains three integers, d_i , p_i , and w_i : the joy delivered by a single decoration on i -th branch, the branch that i -th branch grows from (or 0 if this branch grows directly from the trunk), and the maximum weight of decorations i -th branch can carry ($1 \leq d_i, w_i \leq 10^9$, $0 \leq p_i \leq n$).

It is guaranteed that every branch grows from the trunk, directly or via other branches. Also it is guaranteed that it is possible to use all decorations.

Output

Output one integer: the maximum possible total joy delivered by the decorated baobab.

Example

standard input	standard output
9 6 30 0 4 40 9 2 80 8 3 20 9 2 10 4 3 70 5 8 90 2 4 50 0 6 60 1 3	490

Problem G. Flatland Elections

Input file: *standard input*
Output file: *standard output*
Time limit: 4 seconds
Memory limit: 512 mebibytes

The elections to the parliament of Flatland are taking place soon!

There are N voters in Flatland. For each voter, we know their political preferences. Namely, we know their score on the scale Economic-left–Economic-right and on the scale Libertarian–Authoritarian, and each score is an integer. In this way, each voter is represented by a point on the plane with these two axes. Your party needs to do good in the elections, and for that it needs to get at least K votes.

Before the elections each party needs to present its program. The program is defined by the general line of the party, which is a line on the same plane going through the origin.

Of course, the voters prefer to vote for the party which is closer to their political preferences. The closeness is measured as the distance from the point corresponding to the voter to the line corresponding to the party. You can persuade any voter to vote for your party, but your expenses will be equal to the closeness of this voter to your party.

Your party has emerged very recently, and it does not have a clear political program yet. You are in charge of planning the party's budget. The budget must be sufficiently large to get at least K votes and to cover all expenses, no matter which general line your party will choose. Of course, you want to minimize the budget.

Can you determine the worst possible political program of your party and choose K voters to minimize the total expenses of this program?

Input

The first line of input contains two integers N and K : the number of voters and the number of votes required for your party ($1 \leq K \leq N \leq 2000$).

The following N lines contain coordinates of the voters on the political preference plane, two numbers on each line. The coordinates are integer and do not exceed 10^6 by the absolute value.

Output

On the first line, output the minimum budget. On the second line, describe the worst political program of your party. Namely, output the coordinates of a point lying on the corresponding line which is not equal to the origin. The coordinates must not exceed $2 \cdot 10^9$ by the absolute value. The coordinates and the total expenses may be non-integer.

On the third line, output the space-separated list of K indices of the voters which vote for your party. The voters are numbered in the order of appearance in the input, the order in which you output the indices does not matter.

The output is considered correct if the absolute or relative error does not exceed 10^{-6} by the absolute value, both for the total expenses in the output compared to the correct answer, and for the total expenses in the output compared to the actual expenses for the political program and the list of voters in the output.

Example

standard input	standard output
4 3	12.369316876852980869
3 3	-3.0 12.0
5 4	1 3 4
5 -5	
4 5	

Problem H. Eggs

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

Gosha has an egg tray for 16 eggs on the refrigerator door: two rows of sockets, eight sockets in each row, with an immovable separator in the middle. Initially, there are no eggs in the tray.

When there is at least one egg in the tray, Gosha can take one of them and eat it. When there are at most five eggs in the tray, Gosha can go to a grocery store, buy ten eggs there and place them in the tray. Please note that, after he buys new eggs and placing them in the tray, there is always at least one free socket left.

The eggs will spoil if they are not eaten for too long. So, every time Gosha takes an egg to eat, he should pick one of the oldest eggs in the tray: the ones which were bought earliest. He wants to place the newly bought eggs in such a way that, when he sees the positions of all eggs in the tray, he immediately knows which of them are the oldest. Help Gosha devise a strategy: where to place the eggs he buys and which egg to pick for eating so that the picked egg is one of the oldest.

Input

There are two tests in this problem. In the first test, the first line contains the word “**sample**”, and in the second test the word “**test**”. The answer for the first test is checked for validity only, and on the second test, the strategy is also checked.

Output

Print your strategy as a list of instructions.

Each instruction consists of an initial state (to the left), an action (at the center) and a final state (to the right). Please follow the format shown in the example! The formal output rules are given below the example.

The “**buy**” action must add exactly ten eggs, and the “**eat**” action must remove exactly one egg. A single initial state can occur in the instructions at most twice: at most once with the “**buy**” action and at most once with the “**eat**” action.

In this problem, formatting and the above rules are checked on both tests. However, the strategy checks described below are performed only on the second test.

Gosha uses the strategy as follows. First, he chooses the next possible action: he can eat an egg if there is at least one in the refrigerator, and he can place ten eggs he bought if there are at most five eggs in the refrigerator. After that, Gosha searches for an instruction with the action he chose where the initial state is the actual state of eggs in his refrigerator. Finally, Gosha changes the state to the final state of this instruction.

If there exists a sequence of possible actions such that, eventually, Gosha can not find the instruction he needs, or he picks an egg to eat which is not one of the oldest, the strategy is considered incorrect. If, for every sequence of possible actions, Gosha can find the next instruction, and the egg he eats is always one of the oldest, the strategy is considered correct. Please note that the strategy should account for every sequence of possible actions: when Gosha has two options (eat or buy), both have to be covered.

You are allowed to print instructions with initial states that can not be reached by any sequence of possible actions.

Example

standard input	standard output
sample	**** **** eat **** **** ...* **** **** --- buy ...* ***** **** ---

Note

Here are the rules for printing instructions. Each instruction takes three lines: the first two describe the states and the action, and the third one consists of three “-” characters (minus, ASCII code 45). In each state, empty sockets are denoted by “.” (dot, ASCII code 46), eggs by “*” (asterisk, ASCII code 42), and separators by “|” (vertical line, ASCII code 124). The actions are denoted by words “**buy**” (add ten new eggs) and “**eat**” (remove one egg). On the first line, the action is separated from the states by single spaces, and on the second line, the states are separated by five spaces.

Problem I. Removing Pairs

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

The *pair removal* operation is removing two adjacent characters from a string. For example, given the string “abcd”, we can obtain “ab”, “ad”, and “cd” by pair removal.

You are given a string t . You can apply pair removal any number of times. Is it possible to obtain the string s ?

Input

The first line contains the string s , and the second line contains the string t ($1 \leq |s| \leq |t| \leq 10^5$). Both strings are composed of lowercase letters of English alphabet.

Output

Print “YES” if it is possible to obtain s from t using pair removal, or “NO” otherwise.

Examples

standard input	standard output
axa abcbcxdda	YES
rrr rdfgdfgrdr	NO
w uwwu	NO

Note

In the first test, one of the possible pair removal sequences is the following:

- abcbcxdda
- abcxdda
- abcxa
- axa

Problem J. Boxes on a Shelf

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

Varya has a barn where she stores n boxes with important stuff. She recently nailed another shelf to the barn wall. Now she wants to put as many boxes as possible on the new shelf.

We shall look at the wall with the shelf as a plane. The shelf is then a horizontal segment of length L . There is empty space to the left and to the right of the shelf, which we can consider to be infinite. Box number i is a rectangle with sides a_i and b_i .

A box can stand on the shelf and not fall when one of its sides, or a part of it, lies on the shelf, and the center of the box is above an interior point of the shelf. Boxes can be placed only on the shelf, close to the wall: they can not be placed on top of other boxes or in several layers.

Find is the maximum number of boxes Varya can put on the shelf.

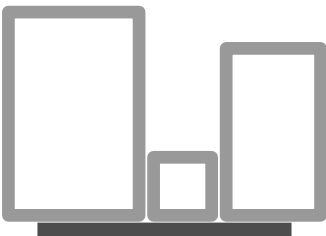

Input

The first line contains an integer n : the number of boxes ($1 \leq n \leq 100\,000$). Each of the following n lines contains two integers a_i and b_i : the sides of the respective box ($1 \leq a_i, b_i \leq 100\,000$). The last line contains an integer L : the length of the shelf ($1 \leq L \leq 10^9$).

Output

Print one integer: the maximum number of boxes Varya can put on the shelf so that they don't fall off.

Examples

standard input	standard output	Notes
4 3 5 6 4 10 20 2 2 7	3	
3 1 1 1 1 1 1 2	2	

Explanations

In the first example, Varya can place three boxes on the shelf: for example, the second box lying on side of length 4 on the left, the fourth box lying on side of length 2 at the center, and the first box lying on side 3 on the right.

In the second example, if all three boxes are placed on the shelf symmetrically, they will fall off, as the centers of left and right boxes will be located exactly above the ends of the shelf.

Problem K. Two Slicers

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

Petya and his friends want to celebrate his birthday party! Petya has a round cake for the occasion. The only thing left is to divide it, and they will have tea.

In order to divide the cake, Petya is going to use slicers. Each slicer has a few blades: a k -slicer is a device which makes k straight cuts from the center of the cake to its boundary, and thus divides the round cake into k identical sectors.

Petya has a red a -slicer and a blue b -slicer. Fortunately, there are exactly $a + b$ friends at the party including Petya. So he decided to use each slicer once, so that the cake will be divided into exactly $a + b$ sectors.

After using the two slicers, the resulting $a + b$ sectors may have different sizes. Nevertheless, the cake should be divided as fairly as possible: the difference between the largest sector and the smallest sector should be the minimum possible.

Find out what is the minimum possible difference that Petya can achieve. Find the difference between the areas of the largest and the smallest sectors after the optimal division. Regard the area of the whole cake as 1. Print the resulting area difference as an irreducible fraction.

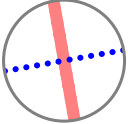
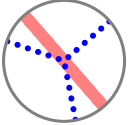
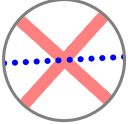
Input

The first line of input contains two space-separated integers a and b : the parameters of red and blue slicers ($2 \leq a, b \leq 100$).

Output

Print an irreducible fraction in the form a / b : the difference between the areas of the largest and the smallest of the $a + b$ resulting sectors after the slicers are applied optimally.

Examples

standard input	standard output	Notes
2 2	0 / 1	
2 3	1 / 4	
4 2	1 / 8	

Explanations

The result of optimal use of slicers is shown to the right of the examples. The cuts made by the red a -slicer are shown as pale red thick lines. The cuts made by the blue b -slicer are shown as blue thin dotted lines.