Problem A. C*++ Calculations

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	64 megabytes

C*++ language is quite similar to C++. The similarity manifests itself in the fact that the programs written in C*++ sometimes behave unpredictably and lead to absolutely unexpected effects. For example, let's imagine an arithmetic expression in C*++ that looks like this (*expression* is the main term):

- $expression ::= summand \mid expression + summand \mid expression - summand$
- $summand ::= increment \mid coefficient{*}increment$
- $increment ::=$ a++ | ++a
- $coefficient ::= 0|1|2|...|1000$

For example, "`5*a++-3*++a+a++`" is a valid expression in C*++.

Thus, we have a sum consisting of several summands divided by signs "`+`" or "`-`". Every summand is an expression "`a++`" or "`++a`" multiplied by some integer coefficient. If the coefficient is omitted, it is suggested being equal to 1.

The calculation of such sum in C*++ goes the following way. First all the summands are calculated one after another, then they are summed by the usual arithmetic rules. If the summand contains "`a++`", then during the calculation first the value of the "`a`" variable is multiplied by the coefficient, then value of "`a`" is increased by 1. If the summand contains "`++a`", then the actions on it are performed in the reverse order: first "`a`" is increased by 1, then — multiplied by the coefficient.

The summands may be calculated in any order, that's why sometimes the result of the calculation is completely unpredictable! Your task is to find its largest possible value.

Input

The first input line contains an integer a ($-1000 \le a \le 1000$) — the initial value of the variable "`a`". The next line contains an expression in C*++ language of the described type. The number of the summands in the expression does not exceed 1000. It is guaranteed that the line describing the expression contains no spaces and tabulation.

Output

Output a single number — the maximal possible value of the expression.

Examples

stdin	stdout

`1`
`5*a++-3*++a+a++`
	`11`

`3`
`a+++++a`
	`8`

Note

Consider the second example. Initially $a = 3$. Suppose that at first the first summand is calculated, and then the second one is. The first summand gets equal to 3, and the value of a is increased by 1. At the calculation of the second summand a is increased once more (gets equal to 5). The value of the second summand is 5, and together they give 8. If we calculate the second summand first and the first summand later, then the both summands equals to 4, and the result is 8, too.

Problem B. Company Income Growth

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	64 megabytes

Petya works as a PR manager for a successful Berland company BerSoft. He needs to prepare a presentation on the company income growth since 2001 (the year of its founding) till now. Petya knows that in 2001 the company income amounted to a_1 billion bourles, in 2002 — to a_2 billion, …, and in the current $(2000 + n)$-th year — a_n billion bourles. On the base of the information Petya decided to show in his presentation the linear progress history which is in his opinion perfect. According to a graph Petya has already made, in the first year BerSoft company income must amount to 1 billion bourles, in the second year — 2 billion bourles etc., each following year the income increases by 1 billion bourles. Unfortunately, the real numbers are different from the perfect ones. Among the numbers a_i can even occur negative ones that are a sign of the company's losses in some years. That is why Petya wants to ignore some data, in other words, cross some numbers a_i from the sequence and leave only some subsequence that has perfect growth.

Thus Petya has to choose a sequence of years y_1, y_2, …, y_k,so that in the year y_1 the company income amounted to 1 billion bourles, in the year y_2 — 2 billion bourles etc., in accordance with the perfect growth dynamics. Help him to choose the longest such sequence.

Input

The first line contains an integer n ($1 \le n \le 100$). The next line contains n integers a_i ($-100 \le a_i \le 100$). The number a_i determines the income of BerSoft company in the $(2000 + i)$-th year. The numbers in the line are separated by spaces.

Output

Output k — the maximum possible length of a perfect sequence. In the next line output the sequence of years y_1, y_2, …, y_k. Separate the numbers by spaces. If the answer is not unique, output any. If no solution exist, output one number 0.

Examples

stdin	stdout

`10`
`-2 1 1 3 2 3 4 -10 -2 5`
	`5`
	`2002 2005 2006 2007 2010`

`3`
`-1 -2 -3`
	`0`

Problem C. Moon Craters

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

There are lots of theories concerning the origin of moon craters. Most scientists stick to the meteorite theory, which says that the craters were formed as a result of celestial bodies colliding with the Moon. The other version is that the craters were parts of volcanoes.

An extraterrestrial intelligence research specialist professor Okulov (the namesake of the Okulov, the author of famous textbooks on programming) put forward an alternate hypothesis. Guess what kind of a hypothesis it was —- sure, the one including extraterrestrial mind involvement. Now the professor is looking for proofs of his hypothesis.

Professor has data from the moon robot that moves linearly in one direction along the Moon surface. The moon craters are circular in form with integer-valued radii. The moon robot records only the craters whose centers lay on his path and sends to the Earth the information on the distance from the centers of the craters to the initial point of its path and on the radii of the craters.

According to the theory of professor Okulov two craters made by an extraterrestrial intelligence for the aims yet unknown either are fully enclosed one in the other or do not intersect at all. Internal or external tangency is acceptable. However the experimental data from the moon robot do not confirm this theory! Nevertheless, professor Okulov is hopeful. He perfectly understands that to create any logical theory one has to ignore some data that are wrong due to faulty measuring (or skillful disguise by the extraterrestrial intelligence that will be sooner or later found by professor Okulov!) That's why Okulov wants to choose among the available crater descriptions the largest set that would satisfy his theory.

Input

The first line has an integer n ($1 \le n \le 2000$) — the number of discovered craters. The next n lines contain crater descriptions in the "c_i r_i" format, where c_i is the coordinate of the center of the crater on the moon robot's path, r_i is the radius of the crater. All the numbers c_i and r_i are positive integers not exceeding 10^9. No two craters coincide.

Output

In the first line output the number of craters in the required largest set. In the next line output space-separated numbers of craters that this set consists of. The craters are numbered from 1 to n in the order in which they were given in the input data. The numbers may be output in any order. If the result is not unique, output any.

Examples

stdin	stdout

4
1 1
2 2
4 1
5 1
	3
	1 2 4

Problem D. Cubical Planet

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	64 megabytes

You can find anything whatsoever in our Galaxy! A cubical planet goes round an icosahedral star. Let us introduce a system of axes so that the edges of the cubical planet are parallel to the coordinate axes and two opposite vertices lay in the points $(0, 0, 0)$ and $(1, 1, 1)$. Two flies live on the planet. At the moment they are sitting on two different vertices of the cubical planet. Your task is to determine whether they see each other or not. The flies see each other when the vertices they occupy lie on the same face of the cube.

Input

The first line contains three space-separated integers (0 or 1) — the coordinates of the first fly, the second line analogously contains the coordinates of the second fly.

Output

Output "YES" (without quotes) if the flies see each other. Otherwise, output "NO".

Examples

stdin	stdout

`0 0 0`
`0 1 0`
	`YES`

`1 1 0`
`0 1 0`
	`YES`

`0 0 0`
`1 1 1`
	`NO`

Problem E. What Has Dirichlet Got to Do with That?

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	64 megabytes

You all know the Dirichlet principle, the point of which is that if n boxes have no less than $n+1$ items, that leads to the existence of a box in which there are at least two items.

Having heard of that principle, but having not mastered the technique of logical thinking, 8 year olds Stas and Masha invented a game. There are a different boxes and b different items, and each turn a player can either add a new box or a new item. The player, after whose turn the number of ways of putting b items into a boxes becomes no less then a certain given number n, loses. All the boxes and items are considered to be different. Boxes may remain empty.

Who loses if both players play optimally and Stas's turn is first?

Input

The only input line has three integers a, b, n ($1 \leq a \leq 10000$, $1 \leq b \leq 30$, $2 \leq n \leq 10^9$) — the initial number of the boxes, the number of the items and the number which constrains the number of ways, respectively. Guaranteed that the initial number of ways is strictly less than n.

Output

Output "Stas" if Masha wins. Output "Masha" if Stas wins. In case of a draw, output "Missing".

Examples

stdin	stdout
`2 2 10`	`Masha`
`5 5 16808`	`Masha`
`3 1 4`	`Stas`
`1 4 10`	`Missing`

Note

In the second example the initial number of ways is equal to 3125.

- If Stas increases the number of boxes, he will lose, as Masha may increase the number of boxes once more during her turn. After that any Stas's move will lead to defeat.
- But if Stas increases the number of items, then any Masha's move will be losing.

Problem F. Pacifist frogs

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	64 megabytes

Thumbelina has had an accident. She has found herself on a little island in the middle of a swamp and wants to get to the shore very much.

One can get to the shore only by hills that are situated along a straight line that connects the little island with the shore. Let us assume that the hills are numbered from 1 to n and the number of a hill is equal to the distance in meters between it and the island. The distance between the n-th hill and the shore is also 1 meter.

Thumbelina is too small to make such jumps. Fortunately, a family of frogs living in the swamp suggests to help her. Each frog agrees to give Thumbelina a ride but Thumbelina should choose only one frog. Each frog has a certain jump length. If Thumbelina agrees to accept help from a frog whose jump length is d, the frog will jump from the island on the hill d, then — on the hill $2d$, then $3d$ and so on until they get to the shore (i.e. find itself beyond the hill n).

However, there is one more problem: mosquitoes also live in the swamp. At the moment they have a siesta, and they are having a nap on some hills. If the frog jumps on a hill with a mosquito the frog will smash it. The frogs Thumbelina has met are pacifists, so they will find the death of each mosquito very much sad. Help Thumbelina choose a frog that will bring her to the shore and smash as small number of mosquitoes as possible.

Input

The first line contains three integers n, m and k ($1 \le n \le 10^9$, $1 \le m, k \le 100$) — the number of hills, frogs and mosquitoes respectively. The second line contains m integers d_i ($1 \le d_i \le 10^9$) — the lengths of the frogs' jumps. The third line contains k integers — the numbers of the hills on which each mosquito is sleeping. No more than one mosquito can sleep on each hill. The numbers in the lines are separated by single spaces.

Output

In the first line output the number of frogs that smash the minimal number of mosquitoes, in the second line — their numbers in increasing order separated by spaces. The frogs are numbered from 1 to m in the order of the jump length given in the input data.

Examples

stdin	stdout

`5 3 5`
`2 3 4`
`1 2 3 4 5`
	`2`
	`2 3`

`1000000000 2 3`
`2 5`
`999999995 999999998 999999996`
	`1`
	`2`

Problem G. Inverse Function

Input file:	`stdin`
Output file:	`stdout`
Time limit:	5 seconds
Memory limit:	64 megabytes

Petya wrote a programme on C++ that calculated a very interesting function $f(n)$. Petya ran the program with a certain value of n and went to the kitchen to have some tea. The history has no records concerning how long the program had been working. By the time Petya returned, it had completed the calculations and had the result. However while Petya was drinking tea, a sly virus managed to destroy the input file so that Petya can't figure out for which value of n the program was run. Help Petya, carry out the inverse function!

Mostly, the program consists of a function in C++ with the following simplified syntax:

- $function$ ::= int f(int n) {*operatorSequence*}
- *operatorSequence* ::= *operator* | *operator operatorSequence*
- *operator* ::= return *arithmExpr*; | if (*logicalExpr*) return *arithmExpr*;
- *logicalExpr* ::= *arithmExpr* > *arithmExpr* | *arithmExpr* < *arithmExpr* | *arithmExpr* == *arithmExpr*
- *arithmExpr* ::= *sum*
- *sum* ::= *product* | *sum* + *product* | *sum* − *product*
- *product* ::= *multiplier* | *product* ∗ *multiplier* | *product*/*multiplier*
- *multiplier* ::= n | *number* | f(*arithmExpr*)
- *number* ::= 0|1|2|...|32767

The whitespaces in a *operatorSequence* are optional.

Thus, we have a function, in which body there are two kinds of operators. There is the operator "return *arithmExpr*;" that returns the value of the expression as the value of the function, and there is the conditional operator "if (*logicalExpr*) return *arithmExpr*;" that returns the value of the arithmetical expression when and only when the logical expression is true. Guaranteed that no other constructions of C++ language — cycles, assignment operators, nested conditional operators etc, and other variables except the n parameter are used in the function. All the constants are integers in the interval [0..32767].

The operators are performed sequentially. After the function has returned a value other operators in the sequence are not performed. Arithmetical expressions are performed taking into consideration the standard priority of the operations. It means that first all the products that are part of the sum are calculated. During the calculation of the products the operations of multiplying and division are performed from the left to the right. Then the summands are summed, and the addition and the subtraction are also performed from the left to the right. Operations ">" (more), "<" (less) and "==" (equals) also have standard meanings.

Now you've got to pay close attention! The program is compiled with the help of 15-bit Berland C++ compiler invented by a Berland company BerSoft, that's why arithmetical operations are performed in a non-standard way. Addition, subtraction and multiplication are performed modulo 32768 (if the result of subtraction is negative, then 32768 is added to it until the number belongs to the interval [0..32767]). Division "/" is a usual integer division where the remainder is omitted.

Examples of arithmetical operations:

$$12345 + 23456 = 3033, \quad 0 - 1 = 32767, \quad 1024 * 1024 = 0, \quad 1000/3 = 333.$$

Guaranteed that for all values of n from 0 to 32767 the given function is performed correctly. That means that:

1. Division by 0 never occures.

2. When performing a function for the value $n = N$ recursive calls of the function f may occur only for the parameter value of $0, 1, \dots, N - 1$. Consequently, the program never has an infinite recursion.

3. As the result of the sequence of the operators, the function always returns a value.

We have to mention that due to all the limitations the value returned by the function f is independent from either global variables or the order of performing the calculations of arithmetical expressions as part of the logical one, or from anything else except the value of n parameter. That's why the f function can be regarded as a function in its mathematical sense, i.e. as a unique correspondence between any value of n from the interval [0..32767] and a value of $f(n)$ from the same interval.

Given the value of $f(n)$, and you should find n. If the suitable n value is not unique, you should find the maximal one (from the interval [0..32767]).

Input

The first line has an integer $f(n)$ from the interval [0..32767]. The next lines have the description of the function f. In the description can be found extra spaces and line breaks (see the examples) which, of course, can't break key words int, if, return and numbers. The size of input data can't exceed 100 bytes.

Output

Output a single number — the answer to the problem. If there's no answer, output "-1" (without quotes).

Examples

stdin	stdout

`17`
`int f(int n)`
`{`
`if (n < 100) return 17;`
`if (n > 99) return 27;`
`}`
	`99`

`13`
`int f(int n)`
`{`
`if (n == 0) return 0;`
`return f(n - 1) + 1;`
`}`
	`13`

`144`
`int f(int n)`
`{`
`if (n == 0) return 0;`
`if (n == 1) return n;`
`return f(n - 1) + f(n - 2);`
`}`
	`24588`

Problem H. Multiplication Table

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	64 megabytes

Petya studies positional notations. He has already learned to add and subtract numbers in the systems of notations with different radices and has moved on to a more complicated action — multiplication. To multiply large numbers one has to learn the multiplication table. Unfortunately, in the second grade students learn only the multiplication table of decimals (and some students even learn it in the first grade). Help Petya make a multiplication table for numbers in the system of notations with the radix k.

Input

The first line contains a single integer k ($2 \le k \le 10$) — the radix of the system.

Output

Output the multiplication table for the system of notations with the radix k. The table must contain $k-1$ rows and $k-1$ columns. The element on the crossing of the i-th row and the j-th column is equal to the product of i and j in the system of notations with the radix k. Each line may have any number of spaces between the numbers (the extra spaces in the samples are put for clarity).

Examples

stdin	stdout

`10`	
	```
	1  2  3  4  5  6  7  8  9
	2  4  6  8 10 12 14 16 18
	3  6  9 12 15 18 21 24 27
	4  8 12 16 20 24 28 32 36
	5 10 15 20 25 30 35 40 45
	6 12 18 24 30 36 42 48 54
	7 14 21 28 35 42 49 56 63
	8 16 24 32 40 48 56 64 72
	9 18 27 36 45 54 63 72 81
	```

`3`	
	```
	1  2
	2 11
	```

Problem I. Tram

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	64 megabytes

In a Berland city S*** there is a tram engine house and only one tram. Three people work in the house — the tram driver, the conductor and the head of the engine house. The tram used to leave the engine house every morning and drove along his loop route. The tram needed exactly c minutes to complete the route. The head of the engine house controlled the tram's movement, going outside every c minutes when the tram drove by the engine house, and the head left the driver without a bonus if he was even one second late.

It used to be so. Afterwards the Berland Federal Budget gave money to make more tramlines in S***, and, as it sometimes happens, the means were used as it was planned. The tramlines were rebuilt and as a result they turned into a huge network. The previous loop route may have been destroyed. S*** has n crossroads and now m tramlines that links the pairs of crossroads. The traffic in Berland is one way so the tram can move along each tramline only in one direction. There may be several tramlines between two crossroads, which go same way or opposite ways. Every tramline links two different crossroads and for each crossroad there is at least one outgoing tramline.

So, the tramlines were built but for some reason nobody gave a thought to increasing the number of trams in S***! The tram continued to ride alone but now the driver had an excellent opportunity to get rid of the unending control of the engine house head. For now due to the tramline network he could choose the route freely! Now at every crossroad the driver can arbitrarily choose the way he can go. The tram may even go to the parts of S*** from where it cannot return due to one way traffic. The driver is not afraid of the challenge: at night, when the city is asleep, he can return to the engine house safely, driving along the tramlines in the opposite direction.

The city people were rejoicing for some of the had been waiting for the tram to appear on their streets for several years. However, the driver's behavior enraged the engine house head. Now he tries to carry out an insidious plan of installing cameras to look after the rebellious tram.

The plan goes as follows. The head of the engine house wants to install cameras at some crossroads, to choose a period of time t and every t minutes turn away from the favourite TV show to check where the tram is. Also the head of the engine house wants at all moments of time, divisible by t, and only at such moments the tram to appear on a crossroad under a camera. There must be a camera on the crossroad by the engine house to prevent possible terrorist attacks on the engine house head. Among all the possible plans the engine house head chooses the plan with the largest possible value of t (as he hates being distracted from his favourite TV show but he has to). If such a plan is not unique, pick the plan that requires the minimal possible number of cameras. Find such a plan.

Input

The first line contains integers n and m ($2 \le n, m \le 10^5$) — the number of crossroads and tramlines in S*** respectively. The next m lines contain the descriptions of the tramlines in "u v" format, where u is the initial tramline crossroad and v is its final crossroad. The crossroads are numbered with integers from 1 to n, and the engine house is at the crossroad number 1.

Output

In the first line output the value of t. In the next line output the value of k — the required number of the cameras. In the next line output space-separated numbers of the crossroads, where the cameras should be installed. Output the numbers in increasing order.

Examples

stdin	stdout

4 5
1 2
2 3
3 4
4 1
1 4
	2
	2
	1 3

Problem J. Spelling Check

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

Petya has noticed that when he types using a keyboard, he often presses extra buttons and adds extra letters to the words. Of course, the spell-checking system underlines the words for him and he has to click every word and choose the right variant. Petya got fed up with correcting his mistakes himself, that's why he decided to invent the function that will correct the words itself. Petya started from analyzing the case that happens to him most of the time, when all one needs is to delete one letter for the word to match a word from the dictionary. Thus, Petya faces one mini-task: he has a printed word and a word from the dictionary, and he should delete one letter from the first word to get the second one. And now the very non-trivial question that Petya faces is: which letter should he delete?

Input

The input data contains two strings, consisting of lower-case Latin letters. The length of each string is from 1 to 10^6 symbols inclusive, the first string contains exactly 1 symbol more than the second one.

Output

In the first line output the number of positions of the symbols in the first string, after the deleting of which the first string becomes identical to the second one. In the second line output space-separated positions of these symbols in increasing order. The positions are numbered starting from 1. If it is impossible to make the first string identical to the second string by deleting one symbol, output one number 0.

Examples

stdin	stdout

`abdrakadabra`
`abrakadabra`
	`1`
	`3`

`aa`
`a`
	`2`
	`1 2`

`competition` `codeforces`	`0`

Problem K. Testing

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	64 megabytes

You take part in the testing of new weapon. For the testing a polygon was created. The polygon is a rectangular field $n \times m$ in size, divided into unit squares 1×1 in size. The polygon contains k objects, each of which is a rectangle with sides, parallel to the polygon sides and entirely occupying several unit squares. The objects don't intersect and don't touch each other.

The principle according to which the weapon works is highly secret. You only know that one can use it to strike any rectangular area whose area is not equal to zero with sides, parallel to the sides of the polygon. The area must completely cover some of the unit squares into which the polygon is divided and it must not touch the other squares. Of course the area mustn't cross the polygon border. Your task is as follows: you should hit no less than one and no more than three rectangular objects. Each object must either lay completely inside the area (in that case it is considered to be hit), or lay completely outside the area.

Find the number of ways of hitting.

Input

The first line has three integers n, m и k ($1 \le n, m \le 1000$, $1 \le k \le 90$) — the sizes of the polygon and the number of objects on it respectively. Next n lines contain m symbols each and describe the polygon. The symbol "`*`" stands for a square occupied an object, whereas the symbol "`.`" stands for an empty space. The symbols "`*`" form exactly k rectangular connected areas that meet the requirements of the task.

Output

Output a single number — the number of different ways to hit a target.

Examples

stdin	stdout

`3 3 3`
`*.*`
`...`
`*..`
	`21`

`4 5 4`
`.*.**`
`...**`
`**...`
`...**`
	`38`

`2 2 1`
`.*`
`..`
	`4`