Problem A. $A + B$

Input file:	`a-plus-b.in`
Output file:	`a-plus-b.out`
Time limit:	1 second
Memory limit:	256 megabytes

Kastus recently got to know about complex numbers. But from all real numbers he was interested only in integers, so he decided to learn only numbers of the form $a + bi$ where $i^2 = -1$, a and b are integers (he had no idea that he was not the first who got interested in these numbers and that they are called Gaussian integers).

It would be nice if Kastus knew how to represent such numbers in computer memory... He is already familiar with some number systems, for example binary and negabinary. He decided that a pair of integers is not an option. One number should look like one number. Kastus had been thinking long and hard what to do, and finally saw an elegant solution: take a number system in base $i - 1$ and the digits 0 and 1. In other words, the expression

$$a + bi = \overline{d_{n-1} \dots d_1 d_0}_{i-1}$$

means that

$$a + bi = (i-1)^{n-1} \cdot d_{n-1} + \dots + (i-1)^1 \cdot d_1 + (i-1)^0 \cdot d_0.$$

For example, $1101_{i-1} = (i-1)^3 + (i-1)^2 + (i-1)^0 = 3$.

Kastus proved that any Gaussian integer can be represented using this notation in the only way (if it has no leading zeros). In addition, he noticed that all the numbers having no more than n significant digits form a dragon curve when he marked them on the complex plane.

Now he is interested in a new simple question: how to make the usual arithmetic operations with the numbers written in this notation. He decided to start with addition, despite the fact that subtraction can not be expressed through addition, whereas addition is expressed through subtraction. But even with this simple question he needs help!

Input

Each of two lines contains one number written in base $i - 1$ with no more than 1 000 000 digits. The numbers don't have leading zeros.

Output

Output the sum of the two given numbers in the same notation.

Examples

a-plus-b.in	a-plus-b.out
1100 1101	111010001

Problem B. Binary Search Tree

Input file:	`binary-tree.in`
Output file:	`binary-tree.out`
Time limit:	1 second
Memory limit:	256 megabytes

A *directed rooted tree* is a weakly connected directed graph all vertices of which have indegree equal to one, except for a *root* that is a vertex of indegree zero. Heads of arcs with tail at vertex v are called *children* of v. A subgraph induced by the set of vertices reachable from v is called a *subtree* of vertex v.

A *binary search tree* is a directed tree that has the following properties. The outdegree of each vertex does not exceed two, besides, one vertex can have no more than one left and no more than one right child. Every vertex v contains a key, moreover, keys of all vertices in the subtree of the left child of v are less than the key of v, and in the subtree of the right child keys are greater than the key of v.

You are given a binary search tree, and there are some vertices chosen to be removed. Please determine if one can end with two trees of different structure depending on the vertex removal order.

The deletion of a vertex v that is not a root is performed as follows. If vertex v has no children, it is simply removed together with incident arc. If vertex v has exactly one child, then v is removed and a pair of its incident arcs is replaced with a new arc that goes from the parent of v to the child of v. If vertex v has two children, it is replaced with a copy of the vertex u that has the least key that exceeds the key of v. The vertex u is then removed from its initial position in the tree according to the rules described above.

Two trees with the same set of keys of vertices have *the same structure* if the subtrees contain the same number of vertices for any pair of vertices with equal keys. Otherwise these trees have different structure.

Input

Input contains several test cases. For each test case the first line contains the number N of vertices in the tree ($1 \le N \le 300\,000$). The ith of the following N lines describe the ith vertex: it contains a removal flag (a character `Y` or `N`—to remove or not), a key (an integer between 1 and 10^9, inclusive) and an index of the parent vertex (that is 0 for root vertex or a number between 1 and $i-1$, inclusive, for any other vertex).

The last line of input contains an integer 0, and this case should not be processed. The sum of N for all test cases in a single file does not exceed 300 000.

It is guaranteed that the keys in any tree are pairwise distinct, the root is not removed and the tree is a binary search tree, i. e., each vertex has no more than one left child (having a smaller key) and no more than one right child (having a greater key), keys in the subtree of the left child are less than the key of the vertex itself, while keys in the subtree of the right child are greater.

Output

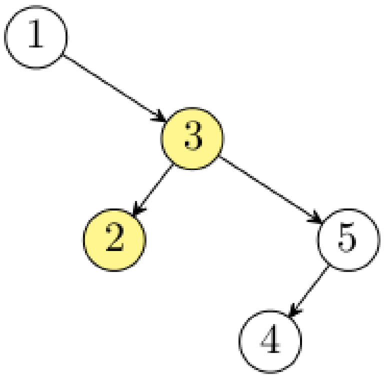
For each test case output a single line `YES` if the structure of the tree depends on the order in which the marked vertices are removed, or `NO` otherwise.

Examples

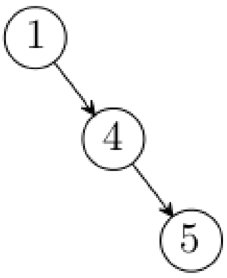
binary-tree.in	binary-tree.out
5	YES
N 1 0	NO
Y 3 1	
Y 2 2	
N 5 2	
N 4 4	
5	
N 1 0	
Y 3 1	
N 2 2	
Y 5 2	
N 4 4	
0	

Note

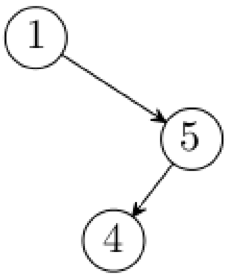
Refer to the figures for clarity.



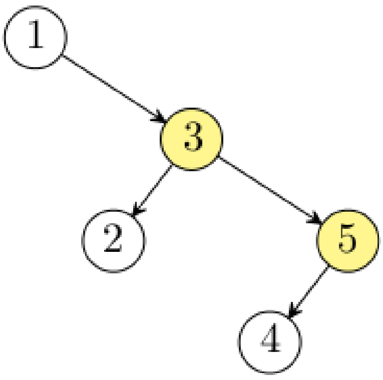
tree for test case #1



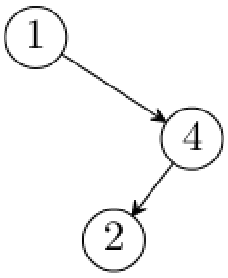
removing 3, then 2



removing 2, then 3



tree for test case #2



result does not depend on removal order

Problem C. Cargo Transportation

Input file:	`cargo.in`
Output file:	`cargo.out`
Time limit:	1 second
Memory limit:	256 megabytes

The shipping company offers two types of vehicles for bulk cargo transportation. A truck of the first type can carry Q_1 tons of goods in one trip. A single trip costs P_1, and the price does not depend on the vehicle loading level. For a truck of the second type these values equal to Q_2 and P_2, respectively.

Find the minimum cost to transfer A tons of cargo.

Input

Input contains five positive integers not greater than a thousand: Q_1, P_1, Q_2, P_2, A. The numbers are separated by spaces.

Output

Output one number: the minimum possible price.

Examples

cargo.in	cargo.out
`3 20 20 100 21`	`120`

Problem D. Delivery

Input file:	`delivery.in`
Output file:	`delivery.out`
Time limit:	1 second
Memory limit:	256 megabytes

Optimum loading of trucks engaged in delivery of goods is a very difficult task. The main difficulty is to fill a vehicle on the way back or to load a not fully filled truck, so the cost of transportation can be different even for the same route in opposite directions.

Vasya took part in programming competitions for more than five years. Today his skills prove useful to him at work. He should select trucks that carry the goods to all cities of the country.

There are N cities numbered from 1 to N in the country. Vasya lives and works in the city with number 1 (the capital). He has to deliver a unit of goods in all other cities from the capital (he can load any number of units into any truck).

Vasya knows about M truck routes. Each truck starts its route in one city and ends in another, without making intermediate stops during the movement. Shipping fees may vary for different routes.

Vasya came up with a heuristic algorithm. Now he wants to determine its efficiency. Help him to determine which routes need to be selected for delivering the goods to all cities (possibly with transshipments between trucks in any of the cities). The total fee for transportation should be minimal.

Input

The first line of the input contains two integers N and M ($2 \le N \le 500$, $1 \le M \le 10\,000$), the number of cities and routes. Each of the next M lines contains three integers x_i, y_i and z_i ($1 \le x_i \ne y_i \le N$, $1 \le z_i \le 10^6$) that specify one truck route from the city with the number x_i to the city with the number y_i, and its cost z_i. There is no pair of the cities with more than one truck route in the same direction.

It is guaranteed that you can select some subset of routes to deliver goods to all the cities.

Output

In the first line output the number of selected routes K ($1 \le K \le M$). In the second line print K different numbers from 1 to M. The routes are numbered in the order they appear in the input.

Separate numbers by spaces.

Examples

`delivery.in`	`delivery.out`

`3 3`
`1 2 5`
`1 3 10`
`3 2 6`
	`2`
	`1 2`

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

Problem E. Frog the Traveller

Input file:	`frog.in`
Output file:	`frog.out`
Time limit:	1 second
Memory limit:	256 megabytes

There are N water lilies arranged in a row in the pond. Frog the Traveller sits on the lily with number 1. She wants to get to the other end (lily with number N) and to visit each of N water lilies exactly once during the journey. The order of the lilies can be any she wants. Or almost any. There are no Geese and Swans on the horizon, so Frog the Traveller has to jump on her own two feet. And this means that from the current water lily she can jump forward or backward on the next, through one or two water lilies.

Frog the Traveller was a good student in school and realizes that if she were omnipotent, she'd have $(N-2)!$ options to travel. She still needs to consider her ability to move. Frog can't even say how many options to travel she has. Help Frog the Traveller!

Additionally, she discovered one issue. Some water lilies had already hidden themselves under the water in anticipation of winter so Frog can't jump here. This detail does not simplify the task and Frog the Traveller is interested in the number of routes taking into account the missing lilies. Of course, neither the distance between other water lilies has changed, nor has the length of the jump increased, so you can assume that lilies hidden in the water are present, but it is forbidden to jump on them.

Input

The first line contains two integers N and K ($2 \le N \le 10^{18}$, $0 \le K \le 100$). The second line contains K distinct integers h_i ($1 < h_i < N$): indices of hidden water lilies in the ascending order. Lilies are numbered starting from 1, and neither the first nor the last are among the hidden ones.

Output

Output the remainder after dividing the number of routes by $10^8 + 2015$.

Examples

frog.in	frog.out
4 0	2
4 1 2	1
3 1 2	1

Problem F. Turning Grille

Input file:	`grille.in`
Output file:	`grille.out`
Time limit:	1 second
Memory limit:	256 megabytes

'Turning grille' method is one of the simplest forms of transposition cipher. A variant of this method is as follows. The sender of the message writes it onto a square sheet of paper gridded into N rows and N columns, one character per cell. The number N is even. A grille is used for ciphering—a pierced sheet of cardboard of the same size as the paper. There are $N^2/4$ holes made in the grille, and one cell can be seen through each hole. When writing a message, the grille is placed on a sheet of paper, and the letters of the message are written into the holes from top to bottom. After all holes have been filled, the grille is turned 90° clockwise, and the writing goes on. The grille can be turned three times, and the message length does not exceed N^2.

The receiver of the message has the same grille and, performing the same manipulations, reads the characters that appear in the holes.

A well-formed grille must not show the same paper sheet cell several times during grille rotations. However, the holes are manufactured by hand and a master can make mistakes, especially when creating a big grille. . .

Write a program that checks if a grille is correct.

Input

The first line of input contains the integer N, the size of the paper ($4 \le N \le 1000$, N is even). N lines follow, each corresponds to a row of the grille and contains N characters `.` (no hole) or `*` (a hole). The number of holes is $N^2/4$.

Output

Output the line `YES` or `NO` depending on whether the grille is well-formed or not.

Examples

<table><tr><th>grille.in</th><th>grille.out</th></tr><tr><td><pre>4
*...
..**
....
.*..</pre></td><td><pre>YES</pre></td></tr><tr><td><pre>4
*...
..**
....
..*.</pre></td><td><pre>NO</pre></td></tr></table>

Problem G. k-palindrome

Input file:	`k-palindrome.in`
Output file:	`k-palindrome.out`
Time limit:	1 second
Memory limit:	256 megabytes

We will say that a string T is a *k-palindrome* for some positive integer k if and only if k is not grater than the length of T and its prefix of the length k is equal to the reversed suffix of that length. For example, `abacaba` is a k-palindrome for any k from 1 to 7 while `abacabada` is only 1-palindrome.

You are given a string S. Try to find the number of k-palindrome substrings of S for all k from 1 to the length of S.

Input

The only line of input contains the string S ($1 \le |S| \le 5000$). The string contains only lowercase letters of the Latin alphabet.

Output

Output $|S|$ integers, the kth of them should be equal to the number of k-palindrome substrings of S.

Examples

`k-palindrome.in`	`k-palindrome.out`
`abacaba`	`14 5 5 2 2 1 1`
`aaaaa`	`15 10 6 3 1`

Problem H. Parallel Worlds

Input file:	`parallel.in`
Output file:	`parallel.out`
Time limit:	1 second
Memory limit:	256 megabytes

You are always told that you are already champions. Well, that is not true. Almost everyone will lose. Only 12 teams in the world are winners. Let us face it.

Yauheni Roizman

Alex is a Baisuralen State University student. Alex and BSU live in two parallel worlds. As we know from school geometry classes, parallel lines do not intersect. However, in reality and unfortunately these two parallel worlds do intersect.

There are some courses in BSU that came from hell. They make parallel worlds intersect and force Alex to visit lectures. Or even more, they cause pain and humiliation. (It was once even said, that the other name for course of 'Functional Analysis' (shortly FUN) is 'Pain and Humiliation'. That is, FUN is not fun.) For example, once Alex slept during such course, and was woken up by professor's voice. Afterwards he was asked if he had moved to Banach Space and was told to move to railway station.

Not everything is so bad, however. There are courses that are from heaven. They are finished in any mark you want without needing to visit them.

You are requested to provide a procedure to establish that some courses are from heaven. As part of that, you need to provide two sets of points P and Q on a plane, each containing N points. All points in P and Q should be distinct, and the intersection of P and Q should be empty. Sets P and Q should satisfy the following property. There should exist N pairwise nonparallel lines, such that sets of projections of P and Q on these lines coincide. Of course, the lines should also be provided. Moreover, pairs of points that coincide are also required.

One can show that for any positive integer N such sets of points exist.

Input

The only line of input contains a single number N ($1 \le N \le 100$).

Output

Output $3 \times N$ lines.

First N lines should contain two real numbers x_i^P y_i^P — coordinates of points in set P.

Next N lines should contain two real numbers x_i^Q y_i^Q — coordinates of points in set Q.

Afterwards output the descriptions of the lines: three real numbers A_i, B_i and C_i—coefficients of the ith line ($A_i x + B_i y + C_i = 0$), and a permutation of numbers from 1 to N ($q_{i1}, q_{i2}, \ldots, q_{iN}$) (the projection of the first point from P should coincide with the projection of the q_{i1}st point of the set Q, the projection of the second point from P should coincide with the projection of the q_{i2}nd point of Q and so on).

Absolute value of all numbers should not be greater than 10^6.

The distance between any two points from $P \cup Q$ should be at least 1. And $P \cap Q = \varnothing$.

Two lines are considered parallel if the angle between the lines is less than 10^{-2} rad., or if the cross product $A_i B_j - B_i A_j$ is less than 10^{-6}.

Two projections coincide if the distance between them is not greater than 10^{-6}.

Examples

parallel.in	parallel.out

1	0 0
	1 0
	1 0 0 1

Problem I. Alien Rectangles

Input file:	`rectangles.in`
Output file:	`rectangles.out`
Time limit:	1 second
Memory limit:	256 megabytes

The spacecraft hovering above the surface of the faraway planet takes pictures of the landscape underneath from time to time. Each photo is a circle centred at the point the spacecraft is orbiting over.

For detailed study of the planet's surface researchers have chosen a Cartesian coordinate system on the photos with the origin placed to the circle's centre. The radius of the circle is R. The following stage of the research requires selecting a region in the form of a nondegenerate rectangle with sides parallel to the coordinate axes and with vertices at the points with integer coordinates. The points should lie inside or on the boundary of the circle.

Please help the researchers and compute the total number of ways to choose a rectangular region. As the number of ways may be big, output it modulo $10^9 + 2015$.

Input

Input contains the only integer R ($1 \le R \le 1\,000\,000$).

Output

Output the only integer: the number of ways to choose a rectangle in the given circle modulo $10^9 + 2015$.

Examples

rectangles.in	rectangles.out
2	9
3	100

Problem J. Restore the Number

Input file:	`sum-of-divisors.in`
Output file:	`sum-of-divisors.out`
Time limit:	1 second
Memory limit:	256 megabytes

Vasya has recently learned that integers can be divided into three classes: perfect, deficient and abundant numbers.

Let n be a positive integer, and let $\sigma(n)$ denote the sum of positive divisors of n (including n itself), i. e.

$$\sigma(n) = \sum_{d|n} d.$$

A *perfect* number is a number that is half the sum of all of its divisors, i. e. $\sigma(n) = 2n$. For example, 28 is perfect, because

$$\sigma(28) = 1 + 2 + 4 + 7 + 14 + 28 = \mathbf{56} = \mathbf{56} = 2 \cdot 28.$$

Numbers where $\sigma(n) < 2n$ are called *deficient*. If, otherwise, $\sigma(n) > 2n$, then the number n is called *abundant*. For instance, 45 is deficient:

$$\sigma(45) = 1 + 3 + 5 + 9 + 15 + 45 = \mathbf{78} < \mathbf{90} = 2 \cdot 45.$$

The number 48 is considered abundant, because

$$\sigma(48) = 1 + 2 + 3 + 4 + 6 + 8 + 12 + 16 + 24 + 48 = \mathbf{124} > \mathbf{96} = 2 \cdot 48.$$

Integer class determination seems to be a simple problem for Vasya. He would like to solve a more advanced problem: given an integer s, find such n that $\sigma(n)$ is equal to s or answer that it is not possible.

Help Vasya and restore an integer using its sum of divisors.

Input

Input contains the only integer s ($1 \le s \le 1\,000\,000\,000$).

Output

Output the integer n such that $\sigma(n) = s$. If there are several solutions, output any of them. If there is no solution, print -1.

Examples

`sum-of-divisors.in`	`sum-of-divisors.out`
3	2
5	-1
7	4

Problem K. UTF-8 Decoder

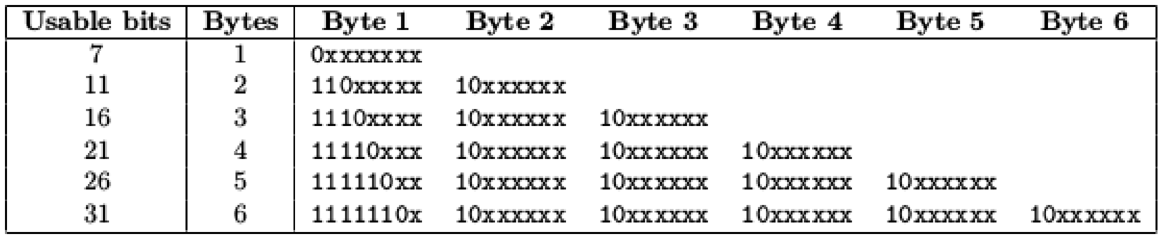
Input file:	`utf-8.in`
Output file:	`utf-8.out`
Time limit:	1 second
Memory limit:	256 megabytes

UTF-8 is a character encoding capable of encoding all possible characters, or code points, in Unicode. Nowadays UTF-8 is the dominant character encoding for the World Wide Web, accounting for 85.1% of all Web pages in September 2015.

Peter works in a large company as a software engineer and develops a new Internet search engine. Its crawler needs a UTF-8 decoder to parse Web pages and put them into index. Peter has already checked if there are any ready-made solutions available. He used his own search engine to look for open-source implementations on the Web and found nothing that satisfied him. Several huge libraries following 'batteries included' philosophy were rejected because they are too heavy and contain tons of code. Several small but relevant libraries didn't get to the top of search results page because Peter's search engine is not perfect at present… So Peter decided to invent the wheel and write his custom lightweight UTF-8 decoder.

Let's define a code point as an integer from range $[0, 2^{31})$. One code point is encoded into variable-length sequence of 8-bit units (bytes).

The design of UTF-8 can be seen in this table (the `x` characters are replaced by the bits of the code point):



Usable bits	Bytes	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
7	1	0xxxxxxx					
11	2	110xxxxx	10xxxxxx				
16	3	1110xxxx	10xxxxxx	10xxxxxx			
21	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx		
26	5	111110xx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	
31	6	1111110x	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx

One-byte codes are used only for the ASCII code point values 0 through 127. In this case the UTF-8 code has the same value as the ASCII code. The high-order bit of these codes is always 0. This means that ASCII text is valid UTF-8.

Code points larger than 127 are represented by multi-byte sequences, composed of a *leading* byte and one or more *continuation* bytes. The leading byte has two or more high-order `1`s followed by a `0`, while continuation bytes all have `10` in the high-order position. UTF-8 offers clear distinction between multi-byte and single-byte characters. The high order bits of every byte determine the type of byte; single bytes (`0xxxxxxx`), leading bytes (`11xxxxxx`), and continuation bytes (`10xxxxxx`) do not share values.

The number of high-order `1`s in the leading byte of a multi-byte sequence indicates the number of bytes in the sequence. The remaining bits of the encoding (the `x` bits in the above patterns) are used for the bits of the code point being encoded, padded with high-order `0`s if necessary. The high-order bits go in the lead byte, lower-order bits in succeeding continuation bytes.

The standard specifies that the correct encoding of a code point use only the minimum number of bytes required to hold the significant bits of the code point. Longer encodings are called overlong and are not valid UTF-8 representations of the code point. This rule maintains a one-to-one correspondence between code points and their valid encodings, so that there is a unique valid encoding for each code point. This ensures that string comparisons and searches are well-defined.

Modern real-life UTF-8 encoding contains more restrictions. For instance, RFC 3629 removed all 5-, 6-byte sequences and some 4-byte sequences in order to match the constraints of the UTF-16 character encoding. Peter wants his decoder to be flexible and to be able to decode as much texts as possible, that's why Peter does not implement these additional restrictions.

You are asked to help Peter and write a UTF-8 decoder.

Input

The first line of input contains an integer N ($1 \leq N \leq 100\,000$). The second line contains the values of N bytes (in range between 0 and 255 each, inclusive) given in hexadecimal. A value consists of two hex digits. The symbols `0`–`9` represent digits zero to nine, and `A`, `B`, `C`, `D`, `E`, `F` represent digits ten to fifteen. Values of bytes are separated by single spaces.

Output

If the input sequence of N bytes can be decoded successfully into sequence of L code points, then in the first line print the number L and in the second line print L code point values (31-bit integers in usual decimal notation with no leading zeros) separated by spaces.

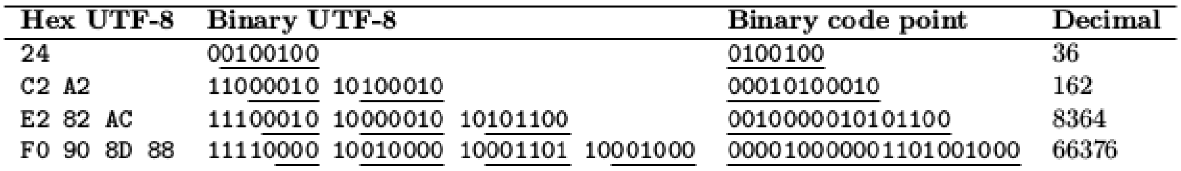
If the input cannot be decoded, output a single line `Epic Fail`.

Examples

utf-8.in	utf-8.out
1 24	1 36
2 C2 A2	1 162
3 E2 82 AC	1 8364
4 F0 90 8D 88	1 66376
4 F0 80 80 AA	Epic Fail
4 00 01 02 03	4 0 1 2 3

Note

Refer to the following table for clarity. Underlined bits are bits from the code point.



Hex UTF-8	Binary UTF-8	Binary code point	Decimal
24	0$\underline{0100100}$	$\underline{0100100}$	36
C2 A2	110$\underline{00010}$ 10$\underline{100010}$	$\underline{00010100010}$	162
E2 82 AC	1110$\underline{0010}$ 10$\underline{000010}$ 10$\underline{101100}$	$\underline{0010000010101100}$	8364
F0 90 8D 88	11110$\underline{000}$ 10$\underline{010000}$ 10$\underline{001101}$ 10$\underline{001000}$	$\underline{000010000001101001000}$	66376

Problem L. Web Programming is marfloW Mathematica

Input file:	`web.in`
Output file:	`web.out`
Time limit:	3 seconds
Memory limit:	256 megabytes

Vtary is yet another professor at Bohemian State Universe (BSU). Vtary likes mathematics, and that is why he always uses a capital letter in this word (Mathematica). He and his colleague Lphic think that all web programming should be done with respect to Mathematica.

Even more than Mathematica Vtary likes psychodelic presentations that hopefully help students to get epilepsy. He even created his own site for publishing presentations. Presentations contain slides about labs, systems and products that help students to think that hell is in reality heaven. Each such slide produces different amount of pain and humiliation.

Alex prefers writing algorithms to reading presentations that cause epilepsy. Thus he and another guy made a contract: Alex writes algorithms for two of them, and the guy works with presentations and labs for two of them. Now Alex wants to calculate his estimated revenue from this deal.

The revenue is calculated as follows. Assume for simplicity that writing an algorithm costs nothing. That is why we are only interested in Vtary's presentations. Each slide in a presentation usually has its own background color, and a slideshow may have a different impact on health and epilepsy. For simplicity, suppose that Vtary's PowerPoint 2003 edition uses a palette of N distinct colors, and Vtary chooses a color for each slide from this set. Different slides of the presentation may have the same color.

Each of the Vtary's presentations consists of L slides. For all possible combinations of N colors on L sildes there is exactly one presentation published on Vtary's website. For any presentaion P, define its weight as $\frac{1}{k+1}$, where k is the amount of colors from N different that do not appear in P. The revenue is calculated as the sum of weights of all presentations of length L.

Input

The only line contains two integers N and L, where $1 \le N \le 10^6$ and $1 \le L \le 10^6$.

Output

Output the revenue in the following format. Suppose that the answer is an irreducible fraction $\frac{A}{B}$. Denote by M the number $10^9 + 2015$. If M is not coprime with B, output '`Mathematica!`' (without quotes), else denote by C the natural number from 1 to M such that $B \cdot C \pmod M = 1$, and output $A \cdot C \pmod M$.

Examples

`web.in`	`web.out`
`2 2`	`3`

Note

In the sample we have four presentations:

1. $(color_1, color_1)$ has weight $\frac{1}{2}$,
2. $(color_1, color_2)$ has weight $\frac{1}{1}$,
3. $(color_2, color_1)$ has weight $\frac{1}{1}$,
4. $(color_2, color_2)$ has weight $\frac{1}{2}$.