Problem A. Assignment Problem

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	256 megabytes

There are m open positions in our company and $n \geq m$ candidates for these positions. We want to hire the best candidates, obviously. We can't hire the same candidate for two or more different positions, so we have to hire exactly m candidates. Let's call the way to choose different candidates for each position *an assignment*. Two assignments are different if there exists a position for which we hire different candidates in these assignments.

There is a matrix A of profits: $A_{ij} \geq 0$ denotes what profit we will gain from hiring j-th candidate for i-th position. We want to maximize the sum of profits we will gain from all hires. An assignment is optimal if it maximizes the sum of profits.

It would be easy to choose the best candidates given the matrix A. Unfortunately, HR world is not so simple, and they can't provide the matrix A for you. Even after interviewing all the candidates we can only compare how two candidates will behave in the same position. More precisely, we know m permutations P_i of length n. For all $1 \leq i \leq m$, $1 \leq x < y \leq n$: $A_{iP_{ix}} > A_{iP_{iy}}$. In human words, for each position we know the ranking of all candidates.

A candidate is *promising* if and only if there exists a matrix A which is consistent with all the given rankings, such that for this matrix there is only one optimal assignment and this particular candidate is hired.

You are to find all promising candidates so that we can conduct more thorough tests with them.

Input

The first line contains two integers n and m ($1 \leq m \leq 11$, $m \leq n \leq 1000$) — the number of candidates and the number of positions.

Next m lines contain rankings for each position. The i-th line contains a permutation $P_{i1}, P_{i2}, \ldots, P_{in}$ of numbers from 1 to n.

Output

In the first line print the number of promising candidates, in the second line print indices of promising candidates **in increasing order**.

Examples

standard input	standard output

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

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

Problem B. Letter Diversity (DIV2 REPLACEMENT)

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

For a string of letters, define the *letter diversity* of the string to be the number of distinct letters in the string. For example, the string "`letter`" has letter diversity 4, and the string "`diversity`" has simplicity 8.

Donald does not likes diversity, so he likes strings which have letter diversity either 1 or 2. You has given him a string as the birthday gift and he asks you to turn it into a string that he likes.

You have a magic eraser which will delete one letter from any string. Compute the minimum number of letters you must erase in order to turn the string into a string with letter diversity at most 2.

Input

The input will consist of a line with a single string consisting of at least 1 and at most 100 lower case letters.

Output

Output a single integer, indicating the minimum number letters you need to erase in order to give the string a letter diversity of 1 or 2.

Examples

standard input	standard output
donald	3
aaa	0

Problem C. Multiple?

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	256 megabytes

Given an integer n, the sequence is called *good* if its elements are from $[1, n]$ and all its non-empty subsequences (not necessarily continuous) have sums not divisible by n.

Calculate the number of good sequences of length $n - k$ modulo 998 244 353.

Input

The only line of input contains two integers n and k ($1 \le k \le n/4 < n < 998\,244\,353$).

Output

Print one number — the answer to the problem.

Examples

standard input	standard output
`4 1`	`2`
`9 2`	`48`
`222222222 222222`	`851798824`

Problem D. Output Limit Exceeded

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

We all know that $\binom{n}{k} = \frac{n\cdot(n-1)\cdot\ldots\cdot(n-k+2)\cdot(n-k+1)}{1\cdot 2\cdot\ldots\cdot(k-1)\cdot k}$ is an integer number for any $0 \le k \le n$. But it would be nice if we could prove it by providing a matching between factors in numerator and denominator, wouldn't it?

Let's build a bipartite graph with k vertices in each part. The i-th vertex in the left part corresponds to factor $(n+1-i)$ from numerator and j-th vertex in the right part corresponds to factor j from denominator. There is an edge i — j if and only if j divides $(n+1-i)$. The number k is *provable* for n if there is a perfect matching in this bipartite graph.

Given n, check if k is provable for each k satisfying $0 \le k \le n$.

Input

The only line contains one integer n ($0 \le n \le 10^{18}$).

Output

Print string of length $(n+1)$ consisting of '`0`' and '`1`', $(k+1)$-th character should be '`1`' if and only if k is provable for n.

What, you think this will get Output Limit Exceeded? Hmmmm... Okay. Let's compress the string.

Let s_0 = "0" and s_1 = "1". You can define $s_2, s_3, \ldots, s_t$. String s_i should be a concatenation of several earlier defined strings. Formally, $\forall\, i\,(2 \le i \le t) : s_i = s_{j_1} + s_{j_2} + \ldots + s_{j_{k_i}}$, here $1 \le k_i$, $\forall\, r\,(1 \le r \le k_i) : j_r < i$. String s_t should be the answer to the problem.

In the first line print one integer t ($2 \le t \le 500$).

In the next $t-1$ lines print the descriptions of s_i. Each description should have a form $k_i\ j_1\ j_2\ \ldots\ j_{k_i}$, with $1 \le k_i$ and $0 \le j_r < i$.

Total length of all descriptions should not exceed 10 000: $\sum_{i=2}^{t} k_i \le 10\,000$.

We can show that for all valid tests there exists a way to construct the answer string abiding all the limitations. If there are several possible ways to do so, print any one of them. Note that you don't have to minimize t or total length of all descriptions.

Examples

standard input	standard output
1	2 2 1 1
0	2 1 1

7	4
	2 1 1
	4 1 2 0 0
	3 3 1 2

Note

In the third sample: $s_2 = s_1 + s_1$ = "1" + "1" = "11", $s_3 = s_1 + s_2 + s_0 + s_0$ = "1" + "11" + "0" + "0" = "11100", $s_4 = s_3 + s_1 + s_2$ = "11100" + "1" + "11" = "11100111",

Problem E. Smol Vertex Cover

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Given an undirected graph, find a minimum vertex cover. Crazy, right?

Let M be the size of maximum matching, and C be the size of minimum vertex cover. If minimum vertex cover is *smol*, which means $C \le M + 1$, then find it.

Input

The first line of input contains two integers n and m ($1 \le n \le 500$, $0 \le m \le \frac{n(n-1)}{2}$) — the number of vertices and edges in the graph.

Next m lines describe edges of the graph, each of them contains two integers u and v ($1 \le u < v \le n$) — vertices connected by an edge. Vertices are numbered from 1 to n.

It is guaranteed that the graph doesn't contain multiple edges.

Output

If minimum vertex cover is smol, then print its size C on the first line, and then C different space-separated vertices that form a vertex cover. Otherwise print "`not smol`" on a single line (without quotes).

If there are several possible smol minimum vertex covers, print any one of them.

Examples

standard input	standard output

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

5 10
1 2
1 3
1 4
1 5
2 3
2 4
2 5
3 4
3 5
4 5
	not smol

3 0	0

Note

Vertex cover is a set of vertices such that for each edge at least one of the endpoints belongs to the set.

Matching is a set of edges such that no two edges from it have common endpoint.

Note that a minimum vertex cover would not be accepted as a correct answer if it is not smol.

Problem F. Triangles (DIV2 REPLACEMENT

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

Check if it is possible to obtain two triangles of given side lengths, by cutting some rectangle with a single line segment, and freely rotating and flipping the resulting pieces.

Input

The input consists of two lines. The first line contains three space-separated integers, indicating the desired side lengths of the first triangle. Similarly, the second line contains three space-separated integers, denoting the desired side lengths of the second triangle. It is guaranteed that the side lengths produce valid triangles. You may assume that the maximum side length of a triangle is 100, and that the minimum is 1.

Output

If there exists a rectangle which could have been cut to form triangles of the given side lengths, output 1. Otherwise, output 0.

Examples

standard input	standard output
12 13 5 13 12 5	1
1 1 1 1 1 1	0

Problem G. Remove the Prime

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	256 megabytes

Two players play a game using an array of positive integers. They make alternating moves, the player who cannot make a move loses. In one move you have to choose a **prime** number p and a non-empty segment $[l; r]$ of the array such that all numbers in this segment are divisible by p, and then remove **all** factors p from each of them. Removing all factors means that we take a number and divide it by p while it is divisible.

Determine who wins if both players play optimally.

Input

The first line contains one integer n ($1 \le n \le 1000$) — the size of array.

The second line contains the array $a_1, a_2, \ldots, a_n$ itself ($1 \le a_i \le 10^{18}$).

Output

Print "`First`" (without quotes) if first player wins and "`Second`" (without quotes) otherwise.

Examples

standard input	standard output
3 2 8 4	First
3 2 12 3	Second

Problem H. Ending by 3 (DIV2 REPLACEMENT)

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

Given an positive integer. Find the smallest base where its representation ends with 3.

For example, decimal number 11 ends with 3 in base 8 (13) and in base 4 (23), so answer will be 4.

Input

Each test case will consist of a single line with a single integer n ($1 \le n < 2^{31}$).

Output

Output a single integer representing the smallest base in which the input n ends with a 3 or -1 if it is impossible to find such a base.

Examples

standard input	standard output
11	4
1	-1

Problem I. Trade

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

It is the last day of your vacation and you decided to buy some memorabilia to remind you about these nice times. There are n merchants, you liked one item from each one. The price written beside the item from i-th merchant is c_i. You have S money with you, and you are ready to spend them on the souvenirs. You don't have any preference so you just want to buy as many different items as possible. It would be an easy job but this is tourist shops we are talking about. They thrive on gullible tourists.

i-th merchant has a persuasion parameter p_i and they are **different** for different merchants. The more souvenirs you already have, the more a merchant is sure about your willingness to spend money on worthless crap. If a merchant sees that you have already bought k souvenirs, he raises the price on his souvenir to $c_i + k \cdot p_i$.

What is the maximal number of souvenirs you can buy?

Input

The first line contains two integers n and S ($1 \le n \le 10^5$, $0 \le S \le 10^9$) — the number of merchants and the amount of money you have.

The second line contains initial prices of all the souvenirs $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le 10^9$).

The third line contains persuasion parameters of all the merchants $p_1, p_2, \ldots, p_n$ ($0 \le p_i \le 10^9$). It is guaranteed that they are distinct.

Output

Print one number — how many souvenirs you can buy.

Examples

standard input	standard output

2 5
1 1
10 11
	1

2 22
10 1
0 10000
	2

1 0
1
0
	0

Problem J. Increasing or Decreasing

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

You are given two permutations A and B of size n. You want to transform A to B in no more than n operations of the following kind:

- Choose a subsegment $[l; r]$ of A and sort it in either increasing or decreasing order.

Note that you don't have to minimize the number of operations, any sequence of operations of length not more than n is ok.

Input

The first line contains one integer n ($1 \le n \le 500$) — the sizes of both permutations.

The second line contains the permutation $A_1, A_2, \ldots, A_n$.

The third line contains the permutation $B_1, B_2, \ldots, B_n$.

Output

On the first line print one integer m ($0 \le m \le n$) — the number of operations.

On the next m lines print the descriptions of operation. One description has a form l_i r_i t_i ($1 \le l_i \le r_i \le n$, t_i is '`I`' or '`D`') and means sort the subsegment $[l_i; r_i]$ in (I)ncreasing or (D)ecreasing order.

If there are different solutions any one will be accepted. It is guaranteed that there is at least one solution.

Examples

standard input	standard output

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

5
5 4 3 2 1
3 2 5 1 4
	4
	2 5 I
	1 4 I
	1 3 D
	3 4 D

5
3 1 4 5 2
3 1 4 5 2
	0

Problem K. Resistance (DIV2 REPLACEMENT)

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Consider the series of numbers where each term is the product of the decimal digits of the previous term. Eventually the term will be reduced to a single digit. For example start with 769:

```
769: 7*6*9 = 378
378: 3*7*8 = 168
168: 1*6*8 = 43
48: 4*8 = 32
32: 3*2 = 6
```

Let's call the number of steps the *resistance* of the number. Given integer X, calculate its resistance.

Input

The input will consist of a single line, with a positive integer of up to 9 decimal digits with no leading zeros.

Output

Output a single integer, representing the resistance of the input number; that is, the number of steps necessary to reduce it to a single digit or -1 if it never will be reduced to single digit.

Examples

standard input	standard output
8	0
2021	1
769	5

Problem L. Extreme Wealth

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

You have a friend working in casino and you decided to use it to your advantage. You'll play Red or Black exactly $R + B$ times and using your insider you know that exactly R times the ball will fall on Red, while all the other B times the ball will fall on Black.

You are starting with initial capital of 1 and before each spin of the roulette you can take any part of your current money and bet it either on Red or Black. You cannot bet on both in one spin. All the money you don't bet remains with you. If you guess the color correctly you get back double the bet, otherwise you lose the bet.

For what maximal X you can **guarantee** you capital to be at least X at the end of the game (if you play optimally)? If $X > 10^9$, you don't really care about the exact value, so just print "`Extreme Wealth`".

Input

The only line contains two integers R and B ($0 \le R, B \le 10^{13}$).

Output

Print the maximal value of X if it is not more than 10^9, otherwise print "`Extreme Wealth`" (without quotes).

Answer "`Extreme Wealth`" will be considered correct if the correct value of X is at least $0.99 \cdot 10^9$. Answer X' will be considered correct if the actual value of X is at most $1.01 \cdot 10^9$ and $\frac{|X'-X|}{X} \le 10^{-6}$.

Examples

standard input	standard output
`3 2`	`3.2000000000000`
`0 29`	`536870912.0000000000000`
`30 0`	`Extreme Wealth`
`37 73`	`5028.4888595832190`

Problem M. Discrete Logarithm is a Joke

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	256 megabytes

$M = 10^{18} + 31$ is a prime number. $g = 42$ is a primitive root modulo M, which means that $g^1 \bmod M, g^2 \bmod M, \ldots, g^{M-1} \bmod M$ are all distinct integers from $[1; M)$. Let's define a function $f(x)$ as the smallest positive integer p such that $g^p = x \bmod M$. f is a bijection from $[1; M)$ to $[1; M)$.

Let's then define a sequence of numbers as follows:

- $a_0 = 960\,002\,411\,612\,632\,915$ (you can copy this number from the sample);
- $a_{i+1} = f(a_i)$.

Given n, find a_n.

Input

The only line of input contains one integer n $(0 \le n \le 10^6)$.

Output

Print a_n.

Examples

standard input	standard output
0	960002411612632915
1	836174947389522544
300300	263358264583736303
1000000	300