Problem A. Assignment Problem

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	256 mebibytes

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. Lockout vs tourist

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

You are playing a lockout match. The rules are simple: it is 1v1 contest, there are n problems with various points assigned, only the first participant to get the problem accepted gets the points, even if the other one is 1 second late they don't get any points. For simplicity we will not include any time limits or early victories: the match continues until each problem is solved by at least one of the participants.

You are pretty good, but... you are facing tourist. It's a disaster, I know. It would be hard to win but you hope to snatch some points to save face. If you and tourist will work on the same problem then he'll beat you, sure. But if you have chosen a different problem then it's your chance! You'll be able to solve this problem before tourist! But then he'll enter the berserk mode and will solve all remaining problems instantly. One problem is better than nothing, isn't it?

Let's formalize the process a bit. Each time you and tourist have some unsolved problems to choose from. If there are no unsolved problems left, then the game ends and you get 0 points. Otherwise, both you and tourist should choose a problem to work on, not knowing which problem the other person chose. If you choose the same problem, tourist solves it before you and we return to the same initial state with fewer problems remaining. Otherwise, you get the points for the problem you have chosen, but the game immediately ends because tourist solves all the remaining problems instantly.

You want to maximize the number of points you'll get, tourist wants to minimize it. What is the expected result of the game if both you and tourist behave optimally?

Input

The first line contains one integer n ($1 \le n \le 22$) — the number of problems.

The second line contains n integers a_i ($1 \le a_i \le 10^9$) — the points for each problem.

Output

Print one number — your expected score.

Your answer is considered correct if its absolute or relative error does not exceed 10^{-6}.

Formally, let your answer be a, and the jury's answer be b. Your answer is accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$.

Examples

standard input	standard output
2 6 7	3.2307692307692
3 1 1 1	0.8333333333333
11 1 2 3 4 5 6 7 8 9 10 11	9.4422713866103

Note

Note that in the first sample tourist could surely win the match always going for the second problem, and against this strategy you could always go for the first problem and get 6 points. But tourist will play optimally **to minimize your expected score** which could sometimes mean allowing you to win the match.

Problem C. Multiple?

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	256 mebibytes

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 mebibytes

We all know that $\binom{n}{k} = \frac{n\cdot(n-1)\cdot\ldots\cdot(n-k+2)\cdot(n-k+1)}{1\cdot2\cdot\ldots\cdot(k-1)\cdot k}$ 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:	2 seconds
Memory limit:	256 mebibytes

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

Each test contains multiple test cases. The first line contains the number of test cases T ($1 \le T \le 10^4$). Description of the test cases follows.

The first line of each test case 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.

It is guaranteed that the sum of n^2 over all test cases does not exceed 250 000.

Output

For each test case, 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 on the second line. 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

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

2
3 0
5 10
1 2
1 3
1 4
1 5
2 3
2 4
2 5
3 4
3 5
4 5
	0
	
	not smol

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. Thanks to MikeMirzayanov

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

... for beautiful Codeforces and Polygon platforms. And thanks to XTX Markets for supporting Codeforces Global Rounds. And also big thanks to **dario2994** who was an author of Codeforces Global Round 11 and who has permitted to reuse his problem `https://codeforces.com/contest/1427/problem/D`. Yep, that's exactly the same problem apart from that doing it in n operations is kinda boring, don't you agree?

You are given a deck of n cards numbered from 1 to n (not necessarily in this order in the deck). You have to sort the deck by repeating the following operation.

Choose $2 \le k \le n$ and split the deck in k nonempty contiguous parts $D_1, D_2, \ldots, D_k$ (D_1 contains the first $|D_1|$ cards of the deck, D_2 contains the following $|D_2|$ cards and so on). Then reverse the order of the parts, transforming the deck into $D_k, D_{k-1}, \ldots, D_2, D_1$ (so, the first $|D_k|$ cards of the new deck are D_k, the following $|D_{k-1}|$ cards are D_{k-1} and so on). The internal order of each packet of cards D_i is unchanged by the operation.

You have to obtain a sorted deck (that is, a deck where the first card is 1, the second is 2 and so on) performing at most 120 operations. It can be proven that it is always possible to sort the deck performing at most 120 operations under the limitations of this problem.

Examples of operation: the following are three examples of valid operations (on three decks with different sizes).

- If the deck is `[3 6 2 1 4 5 7]` (so 3 is the first card and 7 is the last card), we may apply the operation with $k = 4$ and $D_1 =$`[3 6]`, $D_2 =$`[2 1 4]`, $D_3 =$`[5]`, $D_4 =$`[7]`. Doing so, the deck becomes `[7 5 2 1 4 3 6]`.
- If the deck is `[3 1 2]`, we may apply the operation with $k = 3$ and $D_1 =$`[3]`, $D_2 =$`[1]`, $D_3 =$`[2]`. Doing so, the deck becomes `[2 1 3]`.
- If the deck is `[5 1 2 4 3 6]`, we may apply the operation with $k = 2$ and $D_1 =$`[5 1]`, $D_2 =$`[2 4 3 6]`. Doing so, the deck becomes `[2 4 3 6 5 1]`.

Input

The first line of the input contains one integer n ($1 \le n \le 20\,000$) — the number of cards in the deck.

The second line contains n integers $c_1, c_2, \ldots, c_n$ — the cards in the deck. The first card is c_1, the second is c_2 and so on.

It is guaranteed that for all $i = 1, \ldots, n$ there is exactly one $j \in \{1, \ldots, n\}$ such that $c_j = i$.

Output

On the first line, print the number q of operations you perform (it must hold that $0 \le q \le 120$).

Then, print q lines, each describing one operation.

To describe an operation, print on a single line the number k of parts you are going to split the deck in, followed by the sizes of the k parts: $|D_1|, |D_2|, \ldots, |D_k|$.

It must hold that $2 \le k \le n$, and $|D_i| \ge 1$ for all $i = 1, \ldots, k$, and $|D_1| + |D_2| + \ldots + |D_k| = n$.

It can be proven that it is always possible to sort the deck performing at most 120 operations under the limitations of this problem. If there are several ways to sort the deck you can output any one of them. Note that you don't have to minimize q.

Examples

standard input	standard output

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

6 6 5 4 3 2 1	1 6 1 1 1 1 1 1
1 1	0

Problem G. Remove the Prime

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	256 mebibytes

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 of integers $a_1, a_2, \ldots, a_n$ 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. Excluded Min

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	512 mebibytes

Ferume asked me if I can solve this faster than $O(n\sqrt{n}\log n)$. And it turns out I can! Thanks to him for creating this problem and not letting it live with boring solution.

Let S be a multiset containing non-negative integers. You can do the following operation on S arbitrary number of times (possibly zero): choose x such that there are at least two occurrences of x in S, delete one of the occurrences but insert one occurrence of $(x-1)$ or $(x+1)$ instead (you can insert $(x-1)$ only if it is non-negative). Let $F(S)$ be the maximum mex you can achieve with these operations. Here $\text{mex}(S)$ is the minimal non-negative integer which is not present in S.

You are given an array a of length n and q queries $[l; r]$. For each query, find $F(\{a_l, a_{l+1}, \dots, a_r\})$.

Input

The first line contains two integers n, q ($1 \le n, q \le 5 \cdot 10^5$) — the size of array and the number of queries.

The second line contains the array of integers $a_1, a_2, \dots, a_n$ itself ($0 \le a_i \le 5 \cdot 10^5$).

Next q lines contain two integers l_i r_i ($1 \le l_i \le r_i \le n$) — i-th query.

Output

Print answers to queries in the order they are listed in input on separate lines.

Examples

standard input	standard output

3 3
0 0 2
1 3
2 3
3 3
	3
	1
	0

3 2
1 2 2
1 2
1 3
	0
	3

Problem I. Trade

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

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.

The 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 n integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le 10^9$) — initial prices of all the souvenirs.

The third line contains n integers $p_1, p_2, \ldots, p_n$ ($0 \le p_i \le 10^9$) — persuasion parameters of all the merchants. 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 mebibytes

You are given two permutations A and B of size n. Both permutations contain integers from 1 to 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 fine.

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 operations. Each description should be formatted as l_i r_i t_i $(1 \le l_i \le r_i \le n$, t_i is '`I`' or '`D`') and means sorting 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. Rectangle Painting

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	1024 mebibytes

There is a cell grid infinite in left, right, and upwards directions (all the cells with coordinates (x, y) with $x \in \mathbb{Z}$, $y \geq 0$ exist). Initially all the cells are white. You have to process q queries of two types:

1. y_i l_i r_i: paint all cells (x, y_i) for $l_i \leq x \leq r_i$ black. If the cell is already black, its color doesn't change.

2. l_i r_i: consider all cells with x coordinate on the segment $[l_i; r_i]$. Find the highest cell such that all cells exactly under it are black. Formally, you have to find maximal h such that $\exists x \in [l_i; r_i] \forall y \in [0; h)$ cell (x, y) is black.

To enforce processing the queries online they are encrypted using previous answers.

Input

The first line contains one integer q $(1 \leq q \leq 10^5)$ — the number of queries to process.

The next q lines will contain encrypted descriptions of queries. Let S be the sum of answers to all queries of second type processed so far.

Each description is formatted as either "1 $(y_i \oplus S)$ $(l_i \oplus S)$ $(r_i \oplus S)$" or "2 $(l_i \oplus S)$ $(r_i \oplus S)$". It is guaranteed that $0 \leq y_i \leq 2 \cdot 10^5$, $0 \leq l_i \leq r_i \leq 2 \cdot 10^5$. Note that the guarantees are given on parameters after decryption, the numbers in input might not fit in 32-bit integers.

Don't forget to add the new answer to S after each query of the second type.

Output

Print the answers to all queries of the second type on separate lines.

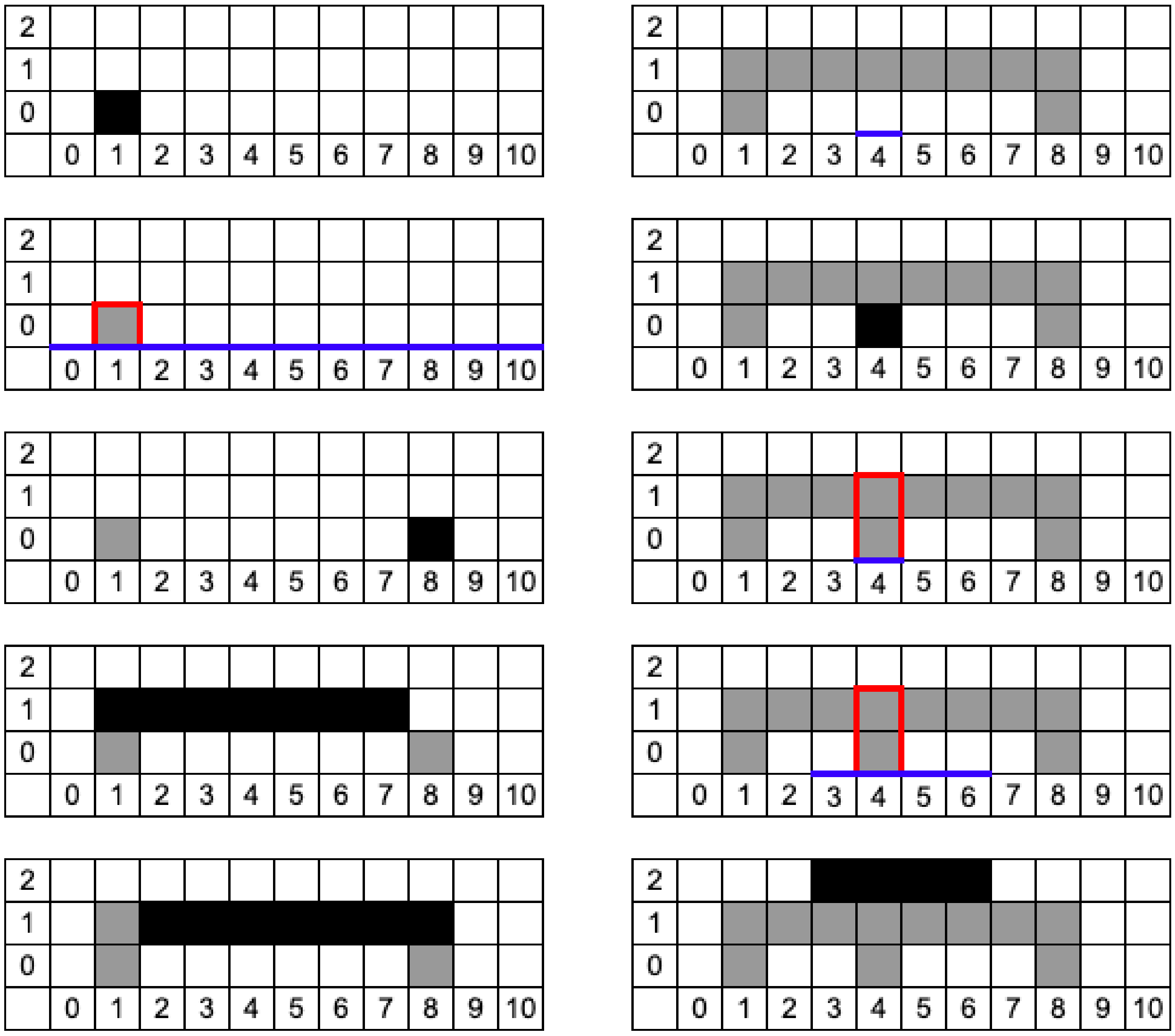
Example

standard input	standard output

10
1 0 1 1
2 0 10
1 1 9 9
1 0 0 6
1 0 3 9
2 5 5
1 1 5 5
2 5 5
2 0 5
1 7 6 3
	1
	0
	2
	2

Note

S	Encrypted	Query	Ans
0	1 0 1 1	1 0 1 1	–
0	2 0 10	2 0 10	1
1	1 1 9 9	1 0 8 8	–
1	1 0 0 6	1 1 1 7	–
1	1 0 3 9	1 1 2 8	–
1	2 5 5	2 4 4	0
1	1 1 5 5	1 0 4 4	–
1	2 5 5	2 4 4	2
3	2 0 5	2 3 6	2
5	1 7 6 3	1 2 3 6	–



Problem L. Extreme Wealth

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	256 mebibytes

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 (not necessarily integer) 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:	5 seconds
Memory limit:	256 mebibytes

Let's take $M = 10^{18} + 31$ which is a prime number, and $g = 42$ which 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 $g^p \equiv x \pmod M$. It is easy to see that 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