Problem A. Circular Shifts

Input file:	`circular.in`
Output file:	`circular.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

RIGS (Research Institute of Given Strings) is a well-known place where people study everything about strings. Vasya works in RIGS for about two years and he already achieved some results. Now he has completed his latest work and he is so amazed about how easy the solution is!

The Big Boss gave Vasya a string… yes, it's the only way to give a task in RIGS. The task was about circular shifts of a given string. Let's define the first (right) circular shift of S as a string which is obtained from S by moving its last character to the beginning of the string. For example, the first circular shift of `abbaab` is `babbaa`. Then, n-th circular shift of S is the first circular shift of $(n-1)$-th one. Vasya needs to consider set of all circular shifts of S… oh, an infinite number of n's… but all these strings have the same length, so there is a chance that the set is finite. And his task was to determine whether this set contains more than two different strings.

Vasya asks you to repeat his achievement so you may have fun too. But be careful, the Boss can be very evil!

Input

The only line of input file contains a non-empty string S. The string contains up to 100 000 lowercase Latin letters.

Output

Output `Yes` if there are more than two different circular shifts of a given string, `No` otherwise.

Examples

circular.in	circular.out
aaaaaaaaaaaaaa	No
aaaaaaaahelpme	Yes
dokittenseatmice	Yes

Problem B. Equation

Input file:	`equation.in`
Output file:	`equation.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Almost all of you have heard about the famous Research Institute of Given Strings — RIGS. But RIGS is only a part of a larger complex of institutes, which work together to achieve better results. The integration between them is very tight, so they have a program called *Staff Exchange*, when people from one institute go work at another one for a few days.

Now Vasya works under this program for the Research Institute of Diophantine Equations (RIDE). They have a common task in collaboration with the Research Institute of Primes (R.I.P.). The task is, given a Diophantine Equation $ax + by = c$, find all its solutions such that x and y are both prime.

Input

The only line of input file contains three integers a, b and c ($0 \leq a, b, c \leq 10^6$).

Output

List all prime solutions of the given Diophantine Equation, in order of increasing x. If there are no solutions, write only one string `NO SOLUTIONS`. If there are more than 10^6 prime solutions, output only one string `TOO MANY`.

Examples

equation.in	equation.out
`2 3 9`	`NO SOLUTIONS`
`2 3 13`	`2 3`

Problem C. Falling Strings

Input file:	`falling.in`
Output file:	`falling.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Vasya is doing research about *falling strings* which are very popular in RIGS today. A falling string behaves in such a way that each second one of its substrings, called *the most heavy substring*, disappears. The most heavy substring is the longest substring of the string such that the first and the last characters of this substring are the same (let's denote this character c), and c never occurs anywhere else in this substring. For example, the most heavy substring of `abacaba` is `bacab`, and for `abc` there are three choices: `a`, `b` and `c`. If there are multiple choices, only the leftmost most heavy substring disappears. It is obvious that any falling string will completely disappear after some small amount of time. For example, `abacaba` will disappear in two seconds, and `abc` will disappear in three.

Your task is to find the number of seconds taken by a given falling string to disappear.

Input

The first line of input contains a non-empty string of lowercase Latin letters. The length of the string does not exceed 5 000 characters.

Output

Output the number of seconds necessary for the string to disappear.

Examples

`falling.in`	`falling.out`
`abacaba`	`2`
`abc`	`3`
`hardexample`	`5`

Problem D. Handled graph

Input file:	`graph.in`
Output file:	`graph.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

As you know, Vasya sometimes works in other institutes. Now he was sent to RIGHT (Research Institute of Graph Handling Theory). And there he... Yes, he handles a graph. Vasya knows how to handle a tree. Of course, you know it too. But Vasya's task is much harder. He should handle a directed graph! Vasya doesn't even know what he should do...

A very clever book says that Vasya should write a number near each vertex (it is called *layer number*) in such a way that any edge starting at vertex with layer number n can go only to vertex with layer number n or $n+1$. The minimal layer number should be exactly 0; moreover, if the maximal layer number is S, all the integers between 0 and S must be used at least once. Of course, there are many ways to handle a graph (for example, put all vertices onto layer 0). But the canonical way is the one which maximizes S.

Your task is to find any of the canonical handling ways of a given graph.

Input

The first line contains two integers n and m ($1 \le n \le 300$, $0 \le m \le 100\,000$) — the number of vertices and the number of edges, respectively. The next m lines contain information about edges. Each of these lines contains two integer numbers a and b ($1 \le a, b \le n$) — numbers of starting and finishing vertices of the corresponding edge. Loops and multi-edges are allowed.

Output

Write one line with N integers; i-th of them should be the layer number of i-th vertex (vertices are numerated as in input).

Examples

graph.in	graph.out
2 1 1 2	0 1

2 2
1 2
2 1
	0 0

Problem E. Isolating Holes

Input file:	`isolating.in`
Output file:	`isolating.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Recently, guys from Research Institute of Cutting and Embedding (RICE) cut n holes in a sheet of paper, which can be considered an infinite 2D plane. The holes are points on this plane. For their next experiment they need to isolate holes by cutting the plane by at most n infinite straight lines in such a way that each hole lies in its own (not necessarily finite) piece.

Vasya needs to do this task. Help him!

Input

The first line of the input contains an integer n ($1 \le n \le 100$). The next n lines contain coordinates of points x_i and y_i. The coordinates are integers not greater than 100 by their absolute values. All points are different.

Output

The first line of output should contain an integer number of lines $k \le n$. The next k lines are the descriptions of straight lines. Each straight line is described by three real numbers a, b and c from its equation $ax + by = c$, $a^2 + b^2$ should be as close as possible to 1, and c should not exceed 10^5 by absolute value. No point from input is allowed to lie within distance of 10^{-6} of any line. Output real numbers as precisely as possible.

If it is impossible to find the plane partition required, output a single -1.

Example

isolating.in	isolating.out

4
0 0
0 1
1 0
1 1
	2
	1 0 0.5
	0 1 0.5

Problem F. k-lindrome

Input file:	`klindrome.in`
Output file:	`klindrome.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Vasya solved many problems about strings during his work in RIGS. But now he faces a problem which is very different from all he saw before.

Vasya's colleague Andrew invented some generalization of palindromes, called *k-lindromes*. He defines them recursively. First, let the result of the function $R_k(S)$ be S for k odd and reversed (i.e. read from back to front) S for k even.

Now let's denote the k-lindrome with base S as $L_k(S)$. Then define $L_1(S) := S$, $L_k(S) := L_{k-1}(S) \cdot R_k(S)$, where $\cdot$ stands for concatenation of two strings (just appending one string to the end of another).

For example, `abbaab` is a 3-lindrome with base `ab`, and `tatata` is not a k-lindrome for any $k > 1$.

Vasya's task is to find the longest substring of the string P which is a k-lindrome for a given k. If there are more than one of them, he needs to find the lexicographically smallest one.

Input

The first line of the input contains a non-empty string P which consists only of lowercase Latin letters. Length of P does not exceed 5 000 characters. The second line contains k ($1 \le k \le 5\,000$).

Output

Output the lexicographically smallest longest k-lindrome substring of P or `NO SOLUTION` if P does not contain substrings which are k-lindromes.

Examples

klindrome.in	klindrome.out
cabbaabc 3	abbaab
tatata 2	NO SOLUTION
avtobusubotva 2	NO SOLUTION

Problem G. Shortest Knight's Path

Input file:	`ndist.in`
Output file:	`ndist.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Research Institute of Stepping Knights (RISK) gives Vasya a **very difficult** problem — find the shortest knight path between two cells on an infinite chess board. Because Vasya doesn't know chess rules, he asked his friend Petr to translate this problem into pure math language. After five minutes of thinking, Petr wrote a fuzzy problem statement:

Given x_1, y_1, x_2, y_2, find $\{\Delta x_i, \Delta y_i, c_i\}_{1 \le i \le k}$ such that:

$$\begin{aligned}
&c_i \ge 0\\
&\Delta x_i \in \{-2, -1, 1, 2\}\\
&\Delta y_i \in \{-2, -1, 1, 2\}\\
&\Delta x_i^2 + \Delta y_i^2 = 5\\
&x_1 + \sum_{1 \le i \le k} c_i \Delta x_i = x_2\\
&y_1 + \sum_{1 \le i \le k} c_i \Delta y_i = y_2\\
&l = \sum_{1 \le i \le k} c_i \to \min
\end{aligned}$$

Input

First line contains a single integer ($1 \le T \le 10$) — number of test cases. Next T lines contain four integers x_1 y_1 x_2 y_2 each ($-10^6 \le x_1, y_1, x_2, y_2 \le 10^6$).

Output

For each test case, the output should contain $(k + 1)$ lines. First of them should contain two integers l and k ($0 \le k \le 8$) separated by a single space. Each of the next k lines should contain three integers Δx_i, Δy_i and c_i separated by single spaces. If there are several answers with the minimal possible value of l, output any of them. Output should **not** contain blank lines between test cases.

Example

ndist.in	ndist.out

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

Problem H. Next, Please...

Input file:	`next.in`
Output file:	`next.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

There was a unique device in the Research Institute of Next Generation (RING). It generates ... well, it generated the next number partition of an integer number n in lexicographical order. If the given partition was the last partition of n, it generated the first partition of $n + 1$.

Unfortunately, somebody recently tried to investigate shockproof characteristics of the device. But after this device was thrown out from the 10th floor window, something strange happened to it. For example, at 10:02 it generated 5 from 3 2. At 10:03 it generated 6 5 from 5 3 3. The scientists of RING think that the device now considers only partitions consisting only of numbers that are not less than some integer m, for example, minutes count.

Vasya's task is to help to check this hypothesis. He needs to write a program for simulating proposed new behavior of the device. More formally, his program should consider an infinite sequence $S(m)$ of number partitions which looks like $P(m)$ $P(m+1)$ $P(m+2)$ ..., where $P(t)$ is lexicographically ordered sequence of all partitions of t consisting of numbers which are not less than m. Here a partition is defined as $t = a_1 + a_2 + \dots a_k$, where $k \geq 1$, and $a_1 \geq a_2 \geq \dots \geq a_k$. Lexicographical order is defined by saying that the first non-equal elements at position p of two partitions a and b (where b occurs later than a in $P(t)$) must satisfy $a_p < b_p$. For example, the start of this sequence for $m = 2$ looks as follows: 2, 3, 2 2, 4, 3 2, 5, 2 2 2, 3 3, 4 2, 6, ...

The task of the program is, given m and an element of $S(m)$, to generate the next one.

Input

The first line of input contains two integers k and m ($1 \leq k \leq 50\,000$, $1 \leq m \leq 50\,000$). The second line consists of k integers a_i ($m \leq a_i \leq 10^6$). It is guaranteed that $a_1 \geq a_2 \geq \dots \geq a_k$.

Output

The first line should contain the length of the next element k'. The second line contains k' numbers forming a non-increasing sequence — the next element. It is guaranteed that k' will never exceed 50 000.

Examples

next.in	next.out

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

`3 3`
`5 3 3`
	`2`
	`6 5`

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

Problem I. Primes Pattern

Input file:	`patprim.in`
Output file:	`patprim.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Research Institute of Primes (R.I.P.) gives Vasya another problem — for given K even positive integers a_i, find N smallest primes p_j such that $p_j + a_i$ is also prime for all $i:\ 1 \le i \le K$.

Input

First line contains two integers K ($1 \le K \le 15$) and N ($1 \le N \le 100$) separated by single space.

Each of next K lines contains single integer a_i ($2 \le a_i \le 30$). Numbers a_i are given in increasing order ($a_{i-1} < a_i$ for all $1 < i \le K$).

Output

Output should contain N lines. The j-th line should contain single integer p_j. You may safely assume that there is always at least one solution such that $p_j < 2^{31} - a_K$.

Examples

patprim.in	patprim.out
1 5 2	3 5 11 17 29

2 1
2
4
	3

Problem J. Polygon

Input file:	`polygon.in`
Output file:	`polygon.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Now Vasya needs to do another work for Research Institute of Cutting and Embedding (RICE). As usual, they just want to cut something, more specifically, a convex polygon. Today the polygon must be cut with a *cross*, which is formed by two infinite straight lines orthogonal to each other. The only condition that needs to hold is that the polygon must be cut into four parts with equal area.

Input

The first line of the input contains an integer n ($3 \le n \le 50\,000$). The next n lines contain two real numbers each — x_i and y_i, coordinates of one vertex of the polygon. It is guaranteed that absolute values of coordinates are less than 1 000. Coordinates are given as precisely as possible. Points are given in counter-clockwise order. There will never be three points on the same straight line.

Output

The first line should contain coordinates of the center of the cross — the point of intersection of these two lines. The second line should contain the coordinates of a vector which is orthogonal to one of the lines. The length of that vector should be as close as possible to 1. All coordinates should be given as precisely as possible, the verifying program will check that the areas are equal with absolute or relative error of 10^{-6}, whichever is greater. If there is no way to cut the given polygon as required, output the word `IMPOSSIBLE`.

Example

polygon.in	polygon.out

4
0.0 0.0
1.0 0.0
1.0 1.0
0.0 1.0
	0.5 0.5
	0 1

Problem K. String Decomposition

Input file:	`strdec.in`
Output file:	`strdec.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

As you remember, Vasya works in RIGS. And now. . . Yes, he faced another problem with strings. The building of RIGS is very big, and Vasya wants to send a string to his friend Petr who works at the other end of the building. However, the sending equipment is rather strange — it cannot deal with strings with lengths greater than k. So Vasya has to divide his string into small blocks.

But there is another problem — the sending equipment asks one coin per string. Of course, Vasya doesn't want to pay too much. Luckily, he knows that the string sending equipment software contains a bug: if Vasya sends two or more equal strings, it will ask for money only for the first time when such a string was sent. This works even if Vasya sends some other strings between the equal ones. Now, Vasya wants to send his string, and he wants to pay as little as possible. The coins are considered equal.

Write a program which will find the minimal number of coins required to send a given string.

Input

The first line contains a non-empty string S (length of S is not greater than 40). The second line contains one integer k ($1 \le k \le 20$). It is guaranteed that S contains only lowercase Latin letters.

Output

Write only one integer — the minimal cost of sending S to Petr.

Examples

strdec.in	strdec.out

`cabbaabc`
`3`
	`3`

`tatata`
`2`
	`1`