

Problem A. Artinals

Input file: `artinals.in`
Output file: `artinals.out`
Time limit: 5 seconds
Memory limit: 64 megabytes

Nick has recently learned about a special kind of sets called *artinian sets* or simply *artinals*. These sets have the advantage of possessing a finite representation, so they can be processed by a computer. However, their formal definition is a bit complicated. Here it is:

- The only *artinal* of height ≤ 0 is the empty set $\emptyset$.
- *Artinals* of height $\leq n$ are exactly the finite sets composed of artinals of height $\leq n - 1$. Here $n \geq 1$ is an arbitrary natural number.
- Finally, A is an *artinal* if A is an artinal of height $\leq n$ for at least one integer n .
- The set of all artinals is denoted by $\mathcal{U}$.

It is immediate from the definition that an artinal of height $\leq n$ is also an artinal of height $\leq n + 1$. Thus for any artinal A we can define its *height* $h(A)$ as the minimal integer n such that A is an artinal of height $\leq n$. An artinal of height n is also called an *n-artinal*.

There were two other definitions which took a lot of time to understand. They are the definition of *canonical order* on $\mathcal{U}$ (denoted by $<$) and the definition of *canonical form* of an artinal:

- The canonical form of an artinal A of height $\leq n$ is a representation $A = \{A_1, A_2, \dots, A_s\}$ where A_i are artinals of height $\leq n - 1$ and $A_1 < A_2 < \dots < A_s$.
- If $A = \{A_1, A_2, \dots, A_s\}$ and $B = \{B_1, B_2, \dots, B_t\}$ are two artinals of height $\leq n$ written in the canonical form, then we put $A < B$ iff there exists an integer k , $1 \leq k \leq \min(s + 1, t)$, such that $A_j = B_j$ for all integer j such that $1 \leq j < k$ and either $k = s + 1$ or $A_k < B_k$.

Now we can define for any artinal A its canonical representation. It is a string $\text{repr}(A)$ composed of characters ‘{’, ‘}’ and ‘,’ defined in the following way: $\text{repr}(\emptyset) = \{\}$, and if A is an artinal with canonical form $A = \{A_1, A_2, \dots, A_s\}$, then $\text{repr}(A) = \{ + \text{repr}(A_1) + , + \dots + , + \text{repr}(A_s) + \}$.

The canonical representation is often rather lengthy. In order to shorten it, the following definition is introduced. For any integer $n \geq 0$ an artinal $\underline{n}$ (called *finite ordinal*) is defined by induction on n : $\underline{0} := \emptyset$ and $\underline{n + 1} := \{\underline{n}\} \cup \underline{n}$ for all $n \geq 0$. Then we can define the *reduced canonical representation* of an artinal in the following way: We take the canonical representation of this artinal and substitute n for any occurrence of the ordinal $\underline{n}$ that is not contained in an occurrence of $\underline{m}$ for some $m > n$.

Then some operations on artinals are defined. These operations (from highest priority to lowest) are:

- Unary intersection $\cap$: for a non-empty artinal $A = \{A_1, A_2, \dots, A_s\}$ put $\cap A := A_1 \cap A_2 \cap \dots \cap A_s$.
- Unary union $\cup$: for any artinal $A = \{A_1, A_2, \dots, A_s\}$ put $\cup A := A_1 \cup A_2 \cup \dots \cup A_s$; $\cup \emptyset := \emptyset$.
- Binary intersection $\cap$: $A \cap B := \{x : x \in A \ \& \ x \in B\}$.
- Binary union $\cup$: $A \cup B := \{x : x \in A \vee x \in B\}$.
- Binary difference $-$: $A - B := \{x \in A : x \notin B\}$.
- Binary symmetrical difference Δ : $A \Delta B := (A - B) \cup (B - A)$.

Besides, some relations between artinals are defined:

- Equality $=$ and inequality $\neq$.
- Inclusion $\subset$ and $\supset$: $A \subset B \Leftrightarrow B \supset A \Leftrightarrow (x \in A \Rightarrow x \in B)$.
- Element relations $\in$ and $\ni$: $B \in A$ (equivalent to $A \ni B$) means that B is an element of A .
- Canonical order relations $<$, $>$, $\leq$, $\geq$ described above (as usual, $A \leq B \Leftrightarrow ((A < B) \vee (A = B))$, $A > B \Leftrightarrow B < A$ and $A \geq B \Leftrightarrow B \leq A$).

Now Nick wants you to write a program that would make some computations with artinals. This program will consist of several operators, each on a separate line. There are five kinds of operators:

- Assignment operator $\langle ident \rangle := \langle expr \rangle$ — sets variable $\langle ident \rangle$ to the value of expression $\langle expr \rangle$.
- Evaluate operator $! \langle expr \rangle$ — evaluates $\langle expr \rangle$ and prints the result in reduced canonical representation on a separate line of output.

- Check condition operator “?” $\langle expr \rangle \langle relation \rangle \langle expr \rangle$ — checks the condition and outputs either “FALSE” or “TRUE” on a separate line of output.
- Comment operator “#” $\langle any_characters \rangle$ — the entire line is copied to the output.
- Empty operator — an empty line (i.e. line consisting only of blank characters) — does nothing.

The following definitions are used:

$\langle ident \rangle ::= \langle alpha \rangle \{ \langle alpha \rangle \}$
 $\langle alpha \rangle ::= \langle letter \rangle | \langle digit \rangle | “_”$
 $\langle digit \rangle ::= “0” | “1” | \dots | “9”$
 $\langle letter \rangle ::= “A” | “B” | \dots | “Z” | “a” | “b” | \dots | “z”$
 $\langle expr \rangle ::= “{” [\langle expr \rangle \{ “,” \langle expr \rangle \}] “}” | \langle ident \rangle | \langle expr \rangle \langle binop \rangle \langle expr \rangle | \langle unop \rangle \langle expr \rangle | “(” \langle expr \rangle “)”$
 $\langle binop \rangle ::= “+” | “*” | “-” | “^”$
 $\langle unop \rangle ::= “+” | “*”$
 $\langle relation \rangle ::= “<” | “>” | “=” | “<=” | “>=” | “<>” | “->” | “<-” | “<<” | “>>”$

The binary operators (in the order they were listed in the definition of $\langle binop \rangle$) correspond to $\cup$, $\cap$, $-$ and Δ ; the unary operators correspond to $\bigcup$ and $\bigcap$; finally, the relations correspond to $<$, $>$, $=$, $\leq$, $\geq$, $\neq$, $\in$, $\ni$, $\subset$, $\supset$. Parentheses “(” and “)” are used to change the precedence of operations as usual. All tokens of input (except several $\langle alpha \rangle$ forming a single $\langle ident \rangle$) can be separated by an arbitrary number of blank characters (i.e. spaces and tabulation characters).

Besides, before the execution of the program the variables with names that are decimal representations (without leading zeros) of non-negative integers $n \leq 2^9$ are set to the finite ordinals $\underline{n}$. All other variables are initialized with $\emptyset$. All identifiers are case-sensitive.

Input

The input file consists of not more than one hundred lines each containing a single operator. No line is longer than 254 characters.

Output

Produce one line of output for each “?”, “!” and “#” operator as described above. It is guaranteed that there will be no “run-time errors” (e.g. unary $\bigcap$ will never be applied to an empty set).

Example

artinals.in	artinals.out
!2 + 2	2
!2*2	2
!3-4	0
# More examples!	# More examples!
	0
00 := 5+3	5
! 3-5	{3,4}
! 00	FALSE
! (5-3)*(5+3)	{1,2,3,7,9}
? 3>9	238
A := {2,3,9}	TRUE
B := {1,7}	TRUE
! A^ B	FALSE
! +239	{1,2,7,{1,7},{1,7}},{1,7},{2,3,9}},{2,3,9}}
? 2->00	
? 2<<00	
? A>>B	
! {{{},{}},{{}},B,{A},{B},{A,B}}+B	

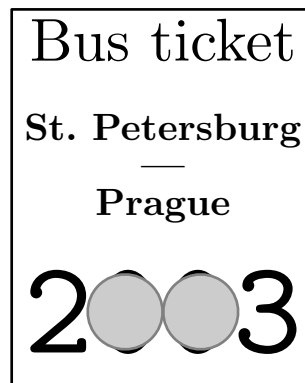
Problem B. Banal Tickets

Input file: `banal.in`
Output file: `banal.out`
Time limit: 2 seconds
Memory limit: 64 megabytes

Peter is fond of number theory. That's why he is looking for interesting bus tickets.

Ticket with the number of length $2N$ is called *interesting* if the product of the first N digits of its number is equal to the product of the last N digits. Other tickets are called *banal*.

Peter has found a used ticket in his pocket. Unfortunately the ticket was punched, so Peter cannot recognize some digits. He wonders whether this ticket was an interesting one. Moreover he wants to know how many different interesting and banal tickets could be punched to get this one.



Help Peter to find answers to his questions.

Input

The first line of the input file contains an integer N ($1 \leq N \leq 18$). The next line contains a string representing the ticket number. If some digit is punched out it is denoted by “?” otherwise it is denoted by itself.

Output

On the first line of the output file print the number of interesting tickets. On the second line print the number of banal tickets.

Example

<code>banal.in</code>	<code>banal.out</code>
2	4
2???3	96
3	46
2?6???3	954

Problem C. Crankshaft

Input file: `cranksft.in`
Output file: `cranksft.out`
Time limit: 2 seconds
Memory limit: 64 megabytes

One company develops a CAD system for crankshaft design. The crankshafts in question are combined of $0 < N \leq 100$ metal plates. All plates are made of the same alloy and have equal (small) thickness. Each plate has the shape of a polygon with $3 \leq V_i \leq 200$ vertices. All plates in the crankshaft are parallel and firmly connected to each other. They are the only elements of the crankshaft.

Among other features, this CAD system should have a routine for determining the rotation axis of the crankshaft. More precisely, customers sometimes need to find a line, perpendicular to the plane of the crankshaft plates, rotation around which causes minimal vibration of the crankshaft. It is well known that such axis must pass through the mass center of the crankshaft.

The crankshaft design is stored as a set of vertex coordinates for each plate in some Cartesian coordinate system. The plane OXY of the system is parallel to the plates surfaces. All the coordinates are integer, $-10^4 \leq X_{ij}, Y_{ij} \leq 10^4$, where $1 \leq j \leq V_i, 1 \leq i \leq N$. The vertices are numbered clockwise (OX axis is directed rightwards, OY axis is directed upwards).

Your program should print the rotation axis coordinates (X_R and Y_R). The correct answer should be given even for a partially designed crankshaft (consisting of two or more non-connected parts). You should suppose that these parts are firmly connected.

Input

The input file consists of integer numbers, delimited by spaces and/or line feeds. The numbers are specified in the following order:

N

$V_1 \ X_{11} \ Y_{11} \ \dots \ X_{1V_1} \ Y_{1V_1}$

...

$V_N \ X_{N1} \ Y_{N1} \ \dots \ X_{NV_N} \ Y_{NV_N}$

Output

The output file should contain two real numbers: X_R and Y_R . These values must be exact to four digits to the right of the decimal point.

Example

<code>cranksft.in</code>	<code>cranksft.out</code>
2 4 -1 -1 -1 1 1 1 1 -1 4 -1 -1 -1 0 0 0 0 -1	-0.1 -0.1
2 3 -2 -1 -2 1 -1 0 3 2 1 2 -1 1 0	0 0

Problem D. Destroying The Graph

Input file: `destroy.in`
Output file: `destroy.out`
Time limit: 2 seconds
Memory limit: 64 megabytes

Alice and Bob play the following game. First, Alice draws some directed graph with N vertices and M arcs. After that Bob tries to destroy it. In a move he may take any vertex of the graph and remove either all arcs incoming into this vertex, or all arcs outgoing from this vertex.

Alice assigns two costs to each vertex: W_i^+ and W_i^- . If Bob removes all arcs incoming into the i -th vertex he pays W_i^+ dollars to Alice, and if he removes outgoing arcs he pays W_i^- dollars.

Find out what minimal sum Bob needs to remove all arcs from the graph.

Input

Input file describes the graph Alice has drawn. The first line of the input file contains N and M ($1 \leq N \leq 100$, $1 \leq M \leq 5000$). The second line contains N integer numbers specifying W_i^+ . The third line defines W_i^- in a similar way. All costs are positive and do not exceed 10^6 . Each of the following M lines contains two integers describing the corresponding arc of the graph. Graph may contain loops and parallel arcs.

Output

On the first line of the output file print W — the minimal sum Bob must have to remove all arcs from the graph. On the second line print K — the number of moves Bob needs to do it. After that print K lines that describe Bob's moves. Each line must first contain the number of the vertex and then '+' or '-' character, separated by one space. Character '+' means that Bob removes all arcs incoming into the specified vertex and '-' that Bob removes all arcs outgoing from the specified vertex.

Example

<code>destroy.in</code>	<code>destroy.out</code>
3 6	5
1 2 3	3
4 2 1	1 +
1 2	2 -
1 1	2 +
3 2	
1 2	
3 1	
2 3	

Problem E. Experiment “X”: Explosions Expected

Input file: `exp.in`
Output file: `exp.out`
Time limit: 2 seconds
Memory limit: 64 megabytes

Unlucky scientist Vasya was tired of permanent mockeries of his colleagues. He built a time machine to go away into the future. But his machine had taken him into the past and then exploded!

Vasya needed some way to earn money! He has taken the job of the Court Alchemist of King Arthur. His first task is to create the philosophers’ stone. Vasya has found K ingredients and now conducts experiments which consist in mixing these ingredients in various proportions. More formally, each experiment consists in taking a_1 ounces of the first ingredient, a_2 ounces of the second and so on. Then all the ingredients are put into the crucible, carefully mixed and heated. All a_i are non-negative integers, and their sum does not exceed S — the capacity of the crucible. In all Vasya’s experiments at least two ingredients are taken, i.e. there are at least two a_i greater than zero.

But all his experiments have been unsuccessful up to now. All mixtures have exploded! The King is unpleased. Maids of honour complain to Queen Guinevere about these awful explosion sounds! The King has decided to give one last chance to Vasya. Tomorrow Vasya is to conduct his next experiment, which could become the last. If the mixture explodes again, Vasya will be executed.

Fortunately, Vasya has just found a way to discard some definitely unsuccessful experiments. He has noted that if for some experiment plan $(a_1, a_2, \dots, a_K)$ the mixture has exploded, it would explode for any plan $(b_1, b_2, \dots, b_K)$ such that $b_i \geq a_i$ for all i . Now Vasya wants to count the number of possible experiments which are not definitely unsuccessful according to his unfortunate experience. He has taken out his pocket computer and now he is writing a program which will compute this number. Can you do this?

Input

The first line of input file contains three positive integers $2 \leq K \leq 30$, $2 \leq S \leq 10000$, $0 \leq M \leq 20$ where M is the number of experiments Vasya has already conducted. The next M lines contain K numbers each, one line per experiment.

Output

Output the number of possible experiment plans for which it is not evident that the mixture would explode.

Example

<code>exp.in</code>	<code>exp.out</code>
2 4 2 1 3 2 1	2

Problem F. Factoring a Polynomial

Input file: `factor.in`
Output file: `factor.out`
Time limit: 2 seconds
Memory limit: 64 megabytes

Recently Georgie has learned about polynomials. A polynomial in one variable can be viewed as a formal sum $a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$, where x is the formal variable and a_i are the coefficients of the polynomial. The greatest i such that $a_i \neq 0$ is called the *degree* of the polynomial. If $a_i = 0$ for all i , the degree of the polynomial is considered to be $-\infty$. If the degree of the polynomial is zero or $-\infty$, it is called *trivial*, otherwise it is called *non-trivial*.

What really impressed Georgie while studying polynomials was the fact that in some cases one can apply different algorithms and techniques developed for integer numbers to polynomials. For example, given two polynomials, one may sum them up, multiply them, or even divide one of them by the other.

The most interesting property of polynomials, at Georgie's point of view, was the fact that a polynomial, just like an integer number, can be factorized. We say that the polynomial is *irreducible* if it cannot be represented as the product of two or more non-trivial polynomials with real coefficients. Otherwise the polynomial is called *reducible*. For example, the polynomial $x^2 - 2x + 1$ is reducible because it can be represented as $(x - 1)(x - 1)$, while the polynomial $x^2 + 1$ is not. It is well known that any polynomial can be represented as the product of one or more irreducible polynomials.

Given a polynomial with integer coefficients, Georgie would like to know whether it is irreducible. Of course, he would also like to know its factorization, but such problem seems to be too difficult for him now, so he just wants to know about reducibility.

Input

The first line of the input file contains n — the degree of the polynomial ($0 \leq n \leq 20$). Next line contains $n + 1$ integer numbers, $a_n, a_{n-1}, \dots, a_1, a_0$ — polynomial coefficients ($-1000 \leq a_i \leq 1000$, $a_n \neq 0$).

Output

Output YES if the polynomial given in the input file is irreducible and NO in the other case.

Example

<code>factor.in</code>	<code>factor.out</code>
2 1 -2 1	NO
2 1 0 1	YES

Problem G. Greatest Common Increasing Subsequence

Input file: `gcis.in`
Output file: `gcis.out`
Time limit: 2 seconds
Memory limit: 64 megabytes

You are given two sequences of integer numbers. Write a program to determine their common increasing subsequence of maximal possible length.

Sequence $S_1, S_2, \dots, S_N$ of length N is called an *increasing subsequence* of a sequence $A_1, A_2, \dots, A_M$ of length M if there exist $1 \leq i_1 < i_2 < \dots < i_N \leq M$ such that $S_j = A_{i_j}$ for all $1 \leq j \leq N$, and $S_j < S_{j+1}$ for all $1 \leq j < N$.

Input

Each sequence is described with M — its length ($1 \leq M \leq 500$) and M integer numbers A_i ($-2^{31} \leq A_i < 2^{31}$) — the sequence itself.

Output

On the first line of the output file print L — the length of the greatest common increasing subsequence of both sequences. On the second line print the subsequence itself. If there are several possible answers, output any of them.

Example

<code>gcis.in</code>	<code>gcis.out</code>
5 1 4 2 5 -12 4 -12 1 2 4	2 1 4

Problem H. Highways

Input file: highways.in
Output file: highways.out
Time limit: 2 seconds
Memory limit: 64 megabytes

In a distant country Lineland there are N cities and they are all located along the highway. The highway is a straight line; it starts from the first city and runs through the second, third city and so on, ending in the N -th city. The i -th city is located at the distance of X_i miles from the first one.

The highway is wide and smooth, so it is a pleasure for all people to drive along it. But there is one problem — all roads in Lineland, including the highway, are one-way. So people are only allowed to drive along the highway from the city with smaller number to the city with greater number and they have to use country roads to get back, and that is not such a great pleasure indeed.

After the new president Mr. Pathwayson was elected in Lineland, he has decided that he would like to make it easier for people to get from one town to another. But he does not dare to change the traditions, and make the highway two-way. Therefore he has decided to build new highways to connect the cities, so that it would be possible to get from any city to any other one by highways. Traditionally, the new highways must be one-way.

Of course, Mr. Pathwayson is a great president, and he wants people to remember him in years. After a thought he has decided that building just one highway would not be enough for that. Therefore he has decided that he must build two new highways. Each highway would connect two different cities. Since people are anxious about their health, and cars running along the highway produce dangerous wastes, each new highway must not pass through any cities, except the cities it connects. Also building two new highways in one city would disturb people too much, so all the cities that would be the ends of the new highways must be different.

You are the assistant of the minister of transportation of Lineland, so you are asked to choose the cities to be connected by the new highways. Since the cost of building a highway is proportional to its length, the total length of the highways must be minimal possible. Write a program to solve this problem. You may assume that the distance between two cities along the new highway is equal to the distance between those cities along the main highway.

Input

The first line of the input file contains N ($2 \leq N \leq 50\,000$).

Next line contains $N - 1$ integer numbers: $X_2, X_3, \dots, X_N$ ($1 \leq X_2 < X_3 < \dots < X_N \leq 10^9$).

Output

If it is impossible to build the highways satisfying all requirements, print number 0 on the first line of the output file.

In the other case on the first line of the output file print the minimal possible total length of the highways to be built. On the second line print S_1 , E_1 , S_2 and E_2 — the numbers of the cities to connect by the first and the second highway, respectively. Note that highways are one-way and must run from S_1 to E_1 and from S_2 to E_2 .

Example

highways.in	highways.out
4	12
3 5 10	3 1 4 2

Problem I. Invariant Polynomials

Input file: `invar.in`
Output file: `invar.out`
Time limit: 2 seconds
Memory limit: 64 megabytes

Consider a real polynomial $P(x, y)$ in two variables. It is called *invariant* with respect to the rotation by an angle α if

$$P(x \cos \alpha - y \sin \alpha, x \sin \alpha + y \cos \alpha) = P(x, y)$$

for all real x and y .

Let's consider the real vector space formed by all polynomials in two variables of degree not greater than d invariant with respect to the rotation by $2\pi/n$. Your task is to calculate the dimension of this vector space.

You might find useful the following remark: Any polynomial of degree not greater than d can be uniquely written in form

$$P(x, y) = \sum_{\substack{i, j \geq 0 \\ i+j \leq d}} a_{ij} x^i y^j$$

for some real coefficients a_{ij} .

Input

The input file contains two positive integers d and n separated by one space. It is guaranteed that they are less than one thousand.

Output

Output a single integer M which is the dimension of the vector space described.

Example

<code>invar.in</code>	<code>invar.out</code>
1 1	3
2 2	4

Problem J. Jogging

Input file: jogging.in
Output file: jogging.out
Time limit: 2 seconds
Memory limit: 64 megabytes

It's Sunday, November 1, 2390, and Eddy has just been elected to the World Council. Of course, it is a very interesting and responsible job, and Eddy is eager to work in the Council, but there is a problem: Eddy is keen on sport. In particular, he likes jogging and he has been jogging at least thirty minutes each day of his life since he was a little boy. Now Eddy will have even less free time than ever before, when he was just a teacher. How will he find spare time for jogging?

Eddy decided that he would jog on his way to his work. Of course, the Council building is rather far away from his house, so he wants to combine jogging with travelling by public transport. Now it has to be told that the moving pathways are the only means of public transport in the Capital. Each line of pathways consists of a pair of straight pathways running in opposite directions with the same speed $v_1 > 0$. These pathways are very long and relatively narrow, so for the purpose of this task they can be considered sharing an infinite straight line on the plane. Besides, for each pathway two numbers are given, T_i^+ and T_i^- — the time necessary for boarding and leaving this pathway. Note that crossing a pathway doesn't take any additional time, but changing from the i -th pathway to the j -th in their intersection point (which in fact is not an intersection since there are special bridges built at these points) takes exactly $T_i^- + T_j^+$ seconds.

Of course, Eddy wants to jog also on the pathways, so he'll move along the pathways with the ground speed $v_1 + v_2$ where $v_2 > 0$ is the speed of Eddy while jogging on still ground.

Now he wants *you* to find a route from his house to the Council building that would require as little time as possible. This route should consist of several segments, some of which can lie on one of the existing pathways.

Input

The first line of input consists of a single integer N , $0 \leq N \leq 50$ — the number of pairs of moving pathways in the city. The second line contains six real numbers $x_1, y_1, x_2, y_2, v_1, v_2$, separated by spaces — the coordinates of Eddy's house and of the Council building, and the speed of pathways and of Eddy respectively. Each of the next N lines contains a description of a pathway consisting of six real numbers $x_{i1}, y_{i1}, x_{i2}, y_{i2}, T_i^+$ and T_i^- , where (x_{i1}, y_{i1}) and (x_{i2}, y_{i2}) are two different points on the pathway and $0 \leq T_i^+, T_i^- \leq 10$ are the boarding and leaving times. All coordinates do not exceed 10000 by their absolute values, and v_1 and v_2 are real numbers ranging from 1 to 100. All pathways lie on different straight lines. Neither (x_1, y_1) nor (x_2, y_2) lie on any of the pathways.

Output

First line of output must contain one real number T — the minimal travel time required. The second line must contain one integer $0 < M \leq 300$ — the number of segments from which the optimal path is composed. Each of the next M lines should consist of one integer $0 \leq k_j \leq N$, the number of the pathway taken (0 means no pathway used), and of the coordinates X_j, Y_j of the end of j -th segment of this path. All real numbers are to be output with at least six digits after decimal point.

Example

jogging.in	jogging.out
2	50.000000
-100 -100 200 100 2.92893219 7.07106781	3
0 0 1 0 0 0	0 0.000000 0.000000
2000 0 2000 1 0 0	1 100.000000 0.000000
	0 200.000000 100.000000

Problem K. Key Insertion

Input file: `key.in`
Output file: `key.out`
Time limit: 2 seconds
Memory limit: 64 megabytes

As an employee of the Macrohard Company, you have been asked to implement the new data structure that would be used to store some integer keys.

The keys must be stored in a special ordered collection that can be considered as an array A , which has an infinite number of locations, numbered starting from 1. Initially all locations are empty. The following operation must be supported by the collection: $Insert(L, K)$, where L is the location in the array and K is some positive integer value.

The operation must be processed as follows:

- If $A[L]$ is empty, set $A[L] \leftarrow K$.
- If $A[L]$ is not empty, perform $Insert(L + 1, A[L])$ and after that set $A[L] \leftarrow K$.

Given N integer numbers $L_1, L_2, \dots, L_N$ you have to output the contents of the array after a sequence of the following operations:

$Insert(L_1, 1)$
 $Insert(L_2, 2)$
 $\dots$
 $Insert(L_N, N)$

Input

The first line of the input file contains N — the number of $Insert$ operations and M — the maximal position that can be used in the $Insert$ operation ($1 \leq N \leq 131\,072$, $1 \leq M \leq 131\,072$).

Next line contains N integer numbers L_i that describe $Insert$ operations to be performed ($1 \leq L_i \leq M$).

Output

Output the contents of the array after a given sequence of $Insert$ operations. On the first line print W — the number of the greatest location that is not empty. After that output W integer numbers — $A[1], A[2], \dots, A[W]$. Output zeroes for empty locations.

Example

key.in	key.out
5 4 3 3 4 1 3	6 4 0 5 2 3 1