Problem A. Belarusian State University

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

Being a student of Belarusian State University (BSU) is an earnest reason for pride. While studying the Theory of Algorithms course, you are obliged to solve many challenging problems before you are admitted to the final exam. Here is one of these problems.

You are given a positive integer n and $4n$ integers $c(i, j, k)$ which can be equal to 0 or 1 ($0 \le i < n$, $j \in \{0, 1\}$, $k \in \{0, 1\}$).

Consider two integers x and y between 0 and $2^n - 1$, inclusively. Let $x = \sum\limits_{i=0}^{n-1} x_i \cdot 2^i$ and $y = \sum\limits_{i=0}^{n-1} y_i \cdot 2^i$ be their binary representations ($x_i, y_j \in \{0, 1\}$). Define $f(x, y) = \sum\limits_{i=0}^{n-1} c(i, x_i, y_i) \cdot 2^i$. Clearly, $f(x, y)$ is also an integer between 0 and $2^n - 1$.

Given two multisets A and B, find the multiset of values $f(a, b)$ over all pairs (a, b), where $a \in A$, $b \in B$.

Input

The first line contains an integer n ($1 \le n \le 18$).

The second line contains n binary strings of 4 digits. The i-th string consists of the values of $c(i-1, 0, 0)$, $c(i-1, 0, 1)$, $c(i-1, 1, 0)$, $c(i-1, 1, 1)$ in this particular order.

The next two lines describe multisets A and B, respectively. The description of a multiset consists of 2^n integers $q_0, q_1, \ldots, q_{2^n-1}$ denoting the quantities of the numbers $0, 1, \ldots, 2^n - 1$ in the multiset ($q_i \ge 0$, $\sum q_i \le 10^9$). There are no other numbers in the multisets.

Output

Print 2^n integers in a single line, the quantities of the numbers $0, 1, \ldots, 2^n - 1$ in the resulting multiset.

Examples

standard input	standard output

`3`
`0111 0110 0001`
`0 0 0 0 0 1 0 0`
`0 0 0 0 0 0 1 0`
	`0 0 0 0 0 0 0 1`

`2`
`1100 1101`
`2 0 2 1`
`2 0 2 1`
	`2 4 3 16`

`1`
`0000`
`142857142 857142857`
`998244353 1755646`
	`999999998000000001 0`

Note

In the first example, you are given 5 and 6. For $x_i, y_i \in \{0, 1\}$, we have

$$f(x_0 + 2x_1 + 4x_2,\ y_0 + 2y_1 + 4y_2) = (x_0 \text{ OR } y_0) + 2 \cdot (x_1 \text{ XOR } y_1) + 4 \cdot (x_2 \text{ AND } y_2).$$

Thus, the only number in the resulting multiset is 7.

Problem B. Beautiful Sequence Unraveling

Input file:	*standard input*
Output file:	*standard output*
Time limit:	7 seconds
Memory limit:	512 mebibytes

You are a happy possessor of the powerful tool called *Beautiful Sequence Unraveler* (BSU). This tool works with beautiful sequences. A *beautiful sequence* is an array $a_1, a_2, \ldots, a_n$ of n integers for which the following statement holds: there are no integers i such that $1 \le i < n$ and $\max\{a_1, \ldots, a_i\} = \min\{a_{i+1}, \ldots a_n\}$.

BSU deals with beautiful sequences pretty well, but you do not know how frequently such sequences occur. So you want to calculate the number of beautiful sequences among all the arrays of length n which consist of integers between 1 and k, inclusively. Since this number may be large, you are required to calculate it modulo prime number p.

Input

The only line contains three integers n, k, p ($1 \le n \le 400$, $1 \le k \le 10^8$, $998\,244\,353 \le p \le 10^9 + 9$).

It is guaranteed that p is prime.

Output

Print the answer to the problem modulo p.

Examples

standard input	standard output
2 2 1000000007	2
3 4 1000000009	36
228 112263 998244353	379700769

Problem C. Brave Seekers of Unicorns

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

You are a member of the Brave Seekers of Unicorns (BSU), the secret magical order. The BSU is fond of seeking unicorns. Recently, they have agreed to call an array $a_1, a_2, \ldots, a_k$ of k integers a *unicorn* if it satisfies the following conditions:

- the array is not empty ($k > 0$);
- there are no three consecutive elements with their bitwise XOR equal to zero ($a_i \oplus a_{i+1} \oplus a_{i+2} \neq 0$ for all $1 \leq i \leq k-2$);
- the array is strictly increasing ($a_i < a_{i+1}$ for all $1 \leq i \leq k-1$);
- the elements of the array are integers between 1 to n, inclusively ($1 \leq a_i \leq n$ for all $1 \leq i \leq k$).

For example, if $n = 10$, then the array $[1, 4, 5, 9]$ is not a unicorn because $1 \oplus 4 \oplus 5 = 0$, but the array $[2, 4, 7, 9]$ is a unicorn.

The Grand Master of the BSU has commanded you to calculate the number of unicorns. Since the number can be pretty large, you must compute it modulo 998 244 353.

Input

The only line contains an integer n ($1 \leq n \leq 10^6$).

Output

Print the number of unicorns modulo 998 244 353.

Examples

standard input	standard output
1	1
2	3
3	6
5	26
322	782852421

Problem D. Bank Security Unification

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

The Bytelandian government has issued the *Bank Security Unification* law (or, shortly, the BSU law). The recent law regulates the usage of Wi-Fi routers in banks and other financial institutions.

According to the BSU law, all the n Wi-Fi routers in a bank must be located in a straight line. Suppose that the i-th router operates at the frequency f_i. Denote the *security* of a connection between two adjacent routers as f_i & f_{i+1}, where & is the bitwise AND operation.

A set of at least two routers numbered $1 \le i_1 < i_2 < \cdots < i_k \le n$ must be chosen as *active*. All other routers will be kept inactive so that they can replace the active ones if any of them would break. Denote the *security of the network* as the sum of the securities of all connections between adjacent active routers. In other words, the security of the network is calculated as $\sum\limits_{j=1}^{k-1} f_{i_j} \ \& \ f_{i_{j+1}}$.

You are an employee of a large Bytelandian bank. Surely, the bank is obliged to comply with the BSU law. The routers are already placed in a line, and their placement cannot be changed. Now you want to choose some of the routers as active to maximize the security of the network.

Input

The first line contains an integer n, denoting the number of Wi-Fi routers in the bank ($2 \le n \le 10^6$).

The second line contains n integers $f_1, f_2, \ldots, f_n$, where f_i is the frequency of the i-th router in the line ($0 \le f_i \le 10^{12}$).

Output

Print the maximum possible security of the network.

Examples

standard input	standard output
5 1 2 3 1 3	5
4 1 2 4 0	0

Problem E. Brief Statements Union

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

Egor learned about the secret organization called *Brief Statements Union* (BSU), whose ultimate goal is to make statements of all competitive programming problems clear and concise, and eliminate those long, boring, and unnecessary tales in the statements.

Egor decided to join the organization. For this purpose, he wrote the following problem with a short statement:

You are given an integer n and k conditions. The i-th condition states that bitwise AND of all integers $a_{l_i}, a_{l_i+1}, \ldots, a_{r_i}$ is equal to x_i.

For each condition i, determine if there exists an array $a_1, a_2, \ldots, a_n$ of n integers which satisfies all the conditions except the condition i. Note that it is OK if the array satisfies the condition i too.

The committee of the organization liked Egor's problem statement. And this is how he got accepted into the organization. Now, Egor has decided to offer the problem to this contest, so you have to solve it.

Input

The first line contains two integers n and k, denoting the required length of the array and the number of the conditions ($1 \le n, k \le 10^6$).

Then k lines follow, the i-th of them contains three integers l_i, r_i, x_i, describing the i-th condition ($1 \le l_i \le r_i \le n$, $0 \le x_i \le 10^{18}$).

Output

Print the binary string of k characters. The i-th character must be equal to '1' if there is an array of length n which satisfies all the conditions with the i-th one being removed. Otherwise, the i-th character must be equal to '0'.

Examples

standard input	standard output

4 3
1 2 1
2 4 3
2 2 1
	011

4 3
1 2 1
3 4 3
2 3 1
	111

Problem F. Border Similarity Undertaking

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	512 mebibytes

There is a large organization called *Border Similarity Undertaking* (BSU) which is located in Bytelandia. The head of the organization has a large map of this glorious country. The map is represented as a matrix A with n rows and m columns. Each element of the matrix is a lowercase Latin letter.

BSU has decided to construct a new factory. The factory may be of any size, but it must be rectangular and its sides must be parallel to the sides of the map. Moreover, as you can deduce from the name of the organization, it is required that all the letters on the border of the rectangle are the same.

The head of BSU hasn't decided where to build a factory yet. So BSU has hired you to calculate the number of possible factory locations.

Formally speaking, you are to find the number of tuples of integers (x_1, y_1, x_2, y_2) such that $1 \le x_1 < x_2 \le n$, $1 \le y_1 < y_2 \le m$, and $A_{i,y_1} = A_{x_1,j} = A_{x_2,j} = A_{i,y_2}$ for all $i \in [x_1, x_2]$, $j \in [y_1, y_2]$.

Input

The first line contains two integers n and m, denoting the number of rows and the number of columns of the map of Bytelandia ($1 \le n, m \le 2000$).

Each of the following n lines contains m lowercase Latin letters, describing the matrix A row by row.

Output

Print the number of possible locations where BSU can construct a new factory.

Examples

standard input	standard output
3 5 zzzzz zxzxz zzzzz	3
4 4 abbc bcca babc acbb	0
12 12 abbabaaaaabb ababaaaaaabb aaabbbbbabbb aabababaaaba abbbaaabaaba baaababbbaba aaaaababbaaa bbabbbbbabaa bbbabbaabaaa aabbbaaaabba babaababbabaa babab aabaaba	25

Problem G. Biological Software Utilities

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

You are developing a software kit named *Biological Software Utilities* (BSU). The kit includes a program that is dedicated to tree recognition. Recall that a *tree* is a connected undirected graph without cycles.

In nature, when a tree grows, two neighboring vertices are added at the same time. Thus, you consider a tree to be *plausible* if, after removing some edges, the resulting graph consists only of connected components with 2 vertices. In other words, a tree is plausible if and only if it has a perfect matching.

Now you are to implement a new function for BSU to calculate the number of plausible trees that have n vertices numbered with distinct integers between 1 and n. Two trees are considered different if there is an edge (u, v) which is present in exactly one of the trees.

Since the number of plausible trees can be very large, you have to calculate it modulo 998 244 353.

Input

The only line contains an integer n, the number of vertices in a tree ($1 \le n \le 10^6$).

Output

Print the number of plausible trees with n vertices modulo 998 244 353.

Examples

standard input	standard output
1	0
2	1
3	0
4	12
7788	178152092

Problem H. Bytelandia States Union

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

There are so many natural wonders in the Bytelandia States Union (BSU)! But the most mysterious wonder is, undoubtedly, the Murbeda Rectangle. Here time and space behave in a rather unusual way. Bytelandian scientists still haven't found a reason why such anomalies occur, even after many years of research. Luckily, they managed to understand how physics works at the Murbeda Rectangle.

Consider the Murbeda Rectangle as a large rectangle. The scientists have divided the rectangle into a grid of $2 \cdot 10^9 \times 2 \cdot 10^9$ small squares. Each square has coordinates (x, y), where the x-axis goes from north to south, and the y-axis goes from west to east. So, the northwestern square is at $(1, 1)$, and the southeastern square is at $(2 \cdot 10^9, 2 \cdot 10^9)$. There is a portal in the square (x_2, y_2) which is the only way to connect the Murbeda Rectangle to the outer world.

Suppose you are in the square (x, y) of the rectangle. You can move in one of four directions (north, south, east, or west), thus increasing or decreasing one of the coordinates by one. You cannot go out of the rectangle: for instance, you cannot go south from the square $(2 \cdot 10^9, 42)$ or go west from the square $(42, 1)$. If you do, you may fall into a deep canyon filled with poisonous snakes.

The most fascinating thing is the amount of time you spend on moving in some direction. If you are in the square (x, y), then:

- going south (increasing the x-coordinate by one) takes $f_s(x, y) = 2xy^2 + 2y^2 + x^2$ seconds;
- going north (decreasing the x-coordinate by one) takes $f_n(x, y) = -2xy^2 + 2y^2 + x^2$ seconds;
- going east (increasing the y-coordinate by one) takes $f_e(x, y) = 2x^2y + 2x^2 + y^2$ seconds;
- going west (decreasing the y-coordinate by one) takes $f_w(x, y) = -2x^2y + 2x^2 + y^2$ seconds.

The amount of time spent on moving between squares may even be negative! The place is *really* special.

The scientists intend to rescue n people from the Murbeda Rectangle. For a person who stands in the square (x_1, y_1), they need to determine the minimum time needed to reach the portal. One can prove that such a minimal amount of time exists, so no one can reach an infinitely small moment by walking around.

Since the place is extremely unusual, each of the n persons may require a different portal.

Input

The first line contains an integer n, the number of people to rescue ($1 \le n \le 5 \cdot 10^4$).

Each of the following n lines contains four integers x_1, y_1, x_2, y_2, denoting the location of the i-th person and the location of their rescue portal ($1 \le x_1, y_1, x_2, y_2 \le 10^9$).

Output

Print n lines. The i-th line should contain the minimal amount of time in seconds for the i-th person to reach their rescue portal. Since this amount can be pretty large, print it modulo 998 244 353.

Example

standard input	standard output

4
1 1 1 1
1 3 2 1
2 1 1 3
10 2 20 6
	0
	12
	17
	16999

Problem I. Binary Supersonic Utahraptors

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Alexey and Boris are playing a game called *Binary Supersonic Utahraptors* (BSU).

Initially, Alexey has n utahraptors, and Boris has m utahraptors. Each utahraptor is either yellow or red.

Then, the players take k turns described by integers $s_1, s_2, \ldots, s_k$. The i-th turn is performed as follows. First, Alexey chooses s_i utahraptors that belong to him and gives them to Boris. Then, Boris chooses s_i utahraptors that belong to him (the utahraptors that Alexey has just given to him may also be chosen) and gives them to Alexey.

When the k moves are done, the score of the game is calculated. The score is equal to $|a_y - b_r|$, where a_y is the number of yellow utahraptors Alexey has, and b_r is the number of red utahraptors Boris has. Alexey's goal is to minimize the score, and Boris wants to maximize it.

Write a program that calculates the score of the game if both players use their optimal strategies.

Input

The first line contains three integers n, m, k, the number of utahraptors obtained by Alexey, the number of utahraptors obtained by Boris, and the number of turns in the game ($1 \le n, m, k \le 3 \cdot 10^5$).

The second line contains n integers a_i, denoting Alexey's utahraptors ($0 \le a_i \le 1$). If $a_i = 0$, then the i-th utahraptor is yellow, otherwise the i-th utahraptor is red.

The third line contains m integers b_i, denoting Boris's utahraptors in the same manner as described above ($0 \le b_i \le 1$).

The fourth line contains k integers s_i, describing the numbers of utahraptors that players give to each other on the i-th turn ($1 \le s_i \le \min\{n, m\}$).

Output

Print the score of the game if both players play optimally.

Example

standard input	standard output

2 3 1
0 0
1 1 1
2
	1

Problem J. Burnished Security Updates

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 mebibytes

Alexander is going to install an important update package called *Burnished Security Updates* (BSU) on his computers. He owns a network which consists of n computers connected by m bidirectional cables.

Eventually, BSU will be installed on every computer in the network. But Alexander doesn't know how the system will behave after the update, so he will first install the update on some non-empty set of computers that satisfies the following conditions:

- no two updated computers are connected directly by a cable;
- each cable must have at least one updated computer as its endpoint;
- the set of the updated computers must be as small as possible.

Formally speaking, if we represent the computer network as a graph, Alexander wants to find an independent set of the graph such that it forms a vertex cover of the same graph. Among all possible sets, he wants to choose one with the least possible size.

Now, you need to help Alexander and find the number of computers on which BSU will be installed. Note that sometimes it can be impossible to find a set satisfying the conditions above at all.

Input

The first line contains two integers n and m, the number of computers and the number of cables ($2 \le n \le 3 \cdot 10^5$, $1 \le m \le 3 \cdot 10^5$).

Each of the following m lines contains two integers x_i and y_i, the endpoints of the i-th cable ($1 \le x_i, y_i \le n$, $x_i \ne y_i$).

It is guaranteed that each pair of computers is connected by no more than one cable.

Output

If there is no such set, print a single integer -1.

Otherwise, print the size of a required set of computers.

Examples

standard input	standard output

4 2
1 2
3 4
	2

4 4
1 2
2 3
3 4
1 4
	2

4 3
1 2
2 3
1 3
	-1

Problem K. Bookcase Solidity United

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	512 mebibytes

You work in *Bookcase Solidity United* (BSU), a company that tests furniture under various load and measures its reliability. Right now, they are testing a bookcase with n shelves, placed from top to bottom.

The bookcase will be tested by putting heavy iridium balls on some of the shelves and observing them break. We assume that all balls are the same and BSU has infinitely many of them.

The engineers measured that the i-th shelf from the top can endure strictly less than a_i balls. If there are $x \ge a_i$ balls on the shelf, it breaks and the balls fall. If there are no unbroken shelves, all the balls fall on the floor. Otherwise, $\lfloor \frac{x}{2} \rfloor$ balls fall on the nearest unbroken shelf j below, and the rest of the balls fall on the floor. (Don't worry, the floor is solid enough to hold all the balls.) If, after this operation, the number of balls on the j-th shelf is not less than a_j, then the j-th shelf breaks, the balls fall from the shelf in the same way as described above, and so on. The process terminates either when all the balls are on the floor or the next shelf is solid enough to hold the balls which fell on it.

To measure the reliability, the employees of BSU put the balls one by one on some of the shelves. The goal is to break k upmost shelves using the minimum possible number of balls. As trying various placement variants is costly, time-consuming, and produces much noise from heavy falling balls, the management of the company has ordered you to calculate the least number of balls to break k upmost shelves, for each k from 1 to n.

Input

The first line contains an integer n, the number of shelves in the bookcase ($1 \le n \le 70$).

The second line contains n integers a_i, where a_i is the minimal number of balls that break the i-th shelf ($1 \le a_i \le 150$). The shelves are numbered from top to bottom.

Output

Print n integers. The k-th integer equals the minimal number of balls needed to break k upmost shelves.

Examples

standard input	standard output
3 8 1 2	8 8 8
5 10 3 3 8 4	10 10 11 17 17

Note

In the first example, we can place eight balls one by one on the first shelf. The shelf will surely break, and $\lfloor \frac{8}{2} \rfloor = 4$ balls will fall on the second shelf. The second shelf now holds $4 > 1$ balls, so it breaks and $\lfloor \frac{4}{2} \rfloor = 2$ balls fall to the third shelf. The third shelf also breaks, and all the balls fall onto the floor. Thus, the answer is 8.

In the second example, to break all the shelves, we can put 2 balls on the third shelf, then we put 3 balls on the second shelf, after that we put 10 balls on the first shelf, and, finally, we put 2 balls on the fourth shelf. So, we put $2 + 3 + 10 + 2 = 17$ balls.

Problem L. Business Semiconductor Units

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Business Semiconductor Units (BSU) is a large international corporation that focuses on selling fast and reliable computers to business clients. Recently, they have decided to develop a new processor model which will work even faster and more reliably than its predecessors.

The R&D department of the company is responsible for designing the instruction set and processor architecture. After the deadline, they should demonstrate the working prototype to the head of the company. Unfortunately, the whole department was playing *Minecraft* most of the time instead of doing their job, so the presented prototype supports only three simple instructions.

Let's take a closer look at their masterpiece. The new processor has 16 registers named from `r0` to `r15`, each of them can store an unsigned 16-bit integer. There is also main memory consisting of $2^{16}+1$ eight-bit cells.

The *program* for this processor is a sequence of instructions. The instructions are executed sequentially, neither jumps nor loops are supported. The processor executes the same sequence of instructions 5000 times. That is, the following procedure is repeated 5000 times: go over the instructions from the start to the end and execute them.

Below you can see the list of available instructions. For clarity, let's call $x \bmod 2^8$ the *lower* part of the number x, and $\left\lfloor \frac{x}{2^8} \right\rfloor$ the *upper* part of the number x. The number in the i-th main memory cell is denoted mem_i.

- `imm r, b`: load the constant number b ($0 \le b < 2^{16}$) into the register named r;
- `ld x, y`: suppose that the register named y stores the number b. Then, the number $mem_{b+1} \cdot 2^8 + mem_b$ is put into the register x;
- `st x, y`: suppose that the register named x stores the number a, and the register y stores the number b. Then, the lower part of b is put into mem_a, and the upper part of b is put into mem_{a+1}.

As you can see, the instruction set is pretty lean, and the R&D department is unsure whether the processor is capable of doing anything non-trivial or not. To make it run some useful programs, they hired you and gave you an assignment. Now, you need to write a program for the new processor that multiplies n non-negative 16-bit numbers modulo 2^{16}.

Input

This problem has no input data.

Output

Output the required program in the following format.

The first line must contain an integer s, the number of instructions in your program ($1 \le s \le 10^5$).

Each of the following s lines must contain a processor instruction. The format of instructions is described above. Be careful and follow the format strictly. All the register names must be valid (that means, from `r0` to `r15`).

Interaction Protocol

Technically, this problem is an output-only interactive problem. [Sounds weird, isn't it? :)]

In each test, the interactor first reads the instructions you wrote to the output. Next, it reads the integer n and n integers a_i from the test ($1 \le n \le 4000$, $0 \le a_i < 2^{16}$). The number n is placed into the register `r0`. For each $1 \le i \le n$, the lower part of a_i is placed into $mem_{2 \cdot i - 2}$, and the upper part is placed into $mem_{2 \cdot i - 1}$. All other registers and memory cells are zeroed initially.

Then, the interactor executes your program. The instructions are performed sequentially, and the execution of the program is performed exactly 5000 times.

After that, the interactor reads your answer from the register `r0` and compares it with $a_1 \cdot a_2 \cdot \ldots \cdot a_n$ modulo 2^{16}.

Example

standard input	standard output

	`4`
	`imm r3, 42`
	`imm r1, 6`
	`st r3, r1`
	`ld r0, r3`

Note

The output in the example does not multiply integers, so submitting this program will give you the "`Wrong Answer`" verdict. The program is only provided to illustrate the output format.

Problem M. Brilliant Sequence of Umbrellas

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Anton has n umbrellas, each of them has a different number from 1 to n written on it. He wants to arrange some of the umbrellas in line so that they would form a *brilliant sequence of umbrellas* (BSU). A sequence of k umbrellas with numbers $a_1, a_2, \ldots, a_k$ is considered a BSU if the following rules apply:

- $a_i > a_{i-1}$ for all $2 \le i \le k$;
- $\gcd(a_i, a_{i-1}) > \gcd(a_{i-1}, a_{i-2})$ for all $3 \le i \le k$. Here, $\gcd(x, y)$ denotes the greatest common divisor of integers x and y.

Anton would like to create a long BSU. Making the longest one doesn't bother him, he thinks that a BSU of length at least $\lceil \frac{2}{3}\sqrt{n} \rceil$ is quite enough.

Anton is busy reading fascinating books about lighthouses, so he asks you to find a BSU that would satisfy him.

Input

The only line contains an integer n, the number of umbrellas ($1 \le n \le 10^{12}$).

Output

The first line should contain an integer k, the length of the BSU you have found ($\lceil \frac{2}{3}\sqrt{n} \rceil \le k \le 10^6$).

The second line should contain k integers a_i, the sequence itself ($1 \le a_i \le n$). The sequence should satisfy the rules mentioned above.

Examples

standard input	standard output
10	3 1 2 6
22	4 1 2 6 15

Note

In the first example, $\lceil \frac{2}{3} \cdot \sqrt{10} \rceil = 3$, $\gcd(2, 4) = 2$, $\gcd(4, 8) = 4$.

In the second example, $\lceil \frac{2}{3} \cdot \sqrt{22} \rceil = 4$, $\gcd(1, 6) = 1$, $\gcd(6, 14) = 2$, $\gcd(14, 21) = 7$.

Problem N. Best Solution Unknown

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 mebibytes

You are the responsible holder of a competition called *Best Solution Unknown* (BSU). The rules of this competition are simple but rather quirky.

First, all the n participants stand in a row. Then, $n - 1$ matches are held. In each match, jury chooses two *adjacent* players. The chosen players are given an NP-hard problem, and they try their best to come up with a good solution. The one who provides a better solution wins a round, the other one leaves the competition. After that, players shift to form a valid row again, so the player adjacent to the player that has left the competition becomes adjacent to the winner of the round. As you can see, after all the $n - 1$ matches, only one player remains, and this player is declared a winner of the competition.

You know the competitors well, so you know the *strength* of each player before the competition. The strength of the i-th player, counting from the left of the row, is a_i. You also know that a player with greater strength wins the match. If the players have equal strength, both have a chance to win. You have noticed that victories motivate the players, so the strength of the winner of a match increases by one.

However, you do not know who plays in each match and who wins a match in case of equal strengths. So, you are wondering who can become the winner of the competition. You thought it was a good problem for the participants of BSU, but, unfortunately, it is not NP-hard, so you have to solve it yourself.

Input

The first line contains an integer n, denoting the number of participants of BSU ($1 \le n \le 10^6$).

The second line contains n integers a_i, where a_i is the initial strength of the i-th participant ($1 \le a_i \le 10^9$).

Output

The first line should contain an integer k, the number of participants that can possibly win the competition ($1 \le k \le n$).

The second line should contain k integers b_i in strictly increasing order, the indices of these participants ($1 \le b_1 < b_2 < \ldots < b_k \le n$).

Examples

standard input	standard output
3 3 2 2	3 1 2 3
3 1 2 1	1 2
5 1 2 3 5 5	3 3 4 5
1 10	1 1