Problem A. Abstract

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

You have a DAG (Directed Acyclic Graph) with n nodes and m edges. The graph has exactly one node x that has no outgoing edges. The i-th node has an integer value a_i in it.

Every second, the following happens:

- For each node i, let $b_i = a_i$.
- For each node i, let $a_i = 0$.
- For each node i, and each node j such that there is an edge from i to j, the value b_i is added to a_j.
- The value $\left\lfloor \frac{b_x}{2} \right\rfloor$ is added to a_x.

Find the first moment of time when all a_i become 0. Since the answer can be very large, output it modulo 998 244 353.

Input

The first line contains two integers n and m ($1 \le n \le 10^4$; $1 \le m \le 10^5$): the number of vertices and edges in the graph.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$): the values in the vertices.

Each of the following m lines contains two integers u and v ($1 \le u, v \le n$) which represent a directed edge from u to v.

It is guaranteed that the graph is a DAG with no multi-edges, and there is exactly one node that has no outgoing edges.

Output

Print a line with a single integer: the first moment of time when all a_i become 0, modulo 998 244 353.

Examples

standard input	standard output
3 2 1 1 1 1 2 2 3	3
6 8 1 1 4 5 1 4 1 4 1 5 2 3 2 5 3 4 4 5 4 6 5 6	8
5 6 7 2 3 6 6 1 2 1 4 2 3 3 4 3 5 4 5	9

Note

Hi, so to me seems like a notorious coincidence. (Codeforces 1704E)

Problem B. Bocchi the Rock

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

Given a circle divided into n arcs by n distinct points lying on it, your task is to determine the number of valid colorings that satisfy certain conditions. Each point on the circle can be colored either red or blue, and each arc can be colored either yellow or pink. Chords can be added between points of the same color, with the restriction that a chord cannot connect a red point and a blue point. No two chords can intersect. Additionally, each point should be connected to at most one chord. Once all chords are added, the circle is divided into various regions by these chords.

A coloring is considered *good* if it is possible to add chords in such a way that each region in the resulting division is *monochrome*. A region is considered *monochrome* if it does not include arcs of both colors. Your goal is to calculate the number of such *good* colorings, taking into account the designated colors of certain points and arcs.

Calculate the number of valid colorings modulo 998 244 353, given the initial colors of some points and arcs.

Input

The first line contains an integer n ($2 \le n \le 5 \cdot 10^4$). The second line contains a string $s = s_1 s_2 \ldots s_{2n}$, representing the color types of the arcs and points.

For every odd i, character s_i can only be "`Y`", "`P`", or "`?`", meaning that the color of $\frac{i+1}{2}$-th arc is yellow, pink, or can be both colors, respectively.

For every even i, character s_i can only be "`R`", "`B`", or "`?`", meaning that the color of $\frac{i}{2}$-th point is red, blue, or can be both colors, respectively.

Output

Print a line with a single integer: the number of valid colorings modulo 998 244 353.

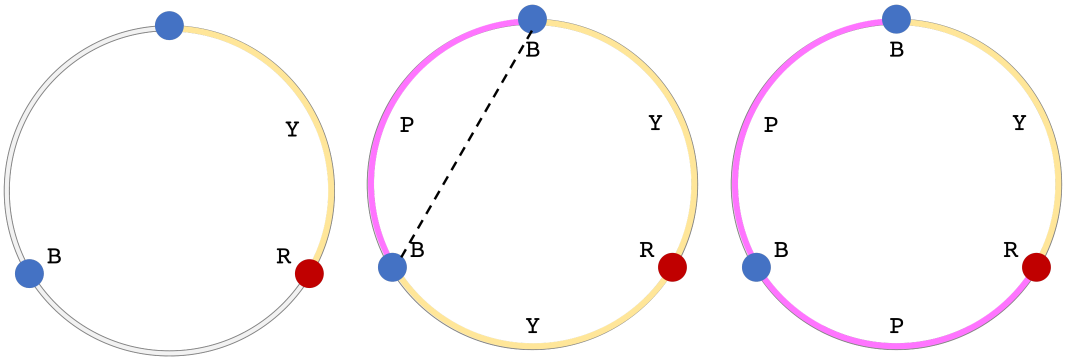
Examples

standard input	*standard output*
2 ????	12
3 ??YR?B	4
5 YBYRPBYRYB	0

Note

In example 1, the 12 valid colorings are: `YBYB`, `YBYR`, `YRYB`, `YRYR`, `PBPB`, `PBPR`, `PRPB`, `PRPR`, `YBPB`, `YRPR`, `PBYB`, `PRYR`.

In example 2, the following figures aid in visual understanding.



The left figure shows the circle before coloring; the middle figure represents a valid coloring as PBYRYB, and we can connect two points colored as B with a chord to prove that the coloring is *good*; the right figure represents an invalid coloring as PBYRPB.

Problem C. Computer Network

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

The additive-increase/multiplicative-decrease (AIMD) algorithm is a feedback control algorithm best known for its use in TCP congestion control. AIMD combines linear growth of the congestion window when there is no congestion with an exponential reduction when congestion is detected. Multiple flows using AIMD congestion control will eventually converge to an equal usage of a shared link. (from Wikipedia)

You are given two arrays of n integers: a and b. You can perform operations on the array a. In one operation, you can let a_i become $a_i + 1$ for all $1 \le i \le n$, or let a_i become $\left\lfloor \frac{a_i}{2} \right\rfloor$ for all $1 \le i \le n$.

Find the minimum number of operations that you have to perform to transform a into b, or determine that it is impossible.

Input

The first line contains an integer n ($1 \le n \le 10^6$).

The second line contains the integer array $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$).

The third line contains the integer array $b_1, b_2, \ldots, b_n$ ($0 \le b_i \le 10^9$).

Output

Print the minimum number of operations needed, or -1 if it's impossible to transform a into b.

Examples

standard input	*standard output*
5 1 2 3 4 5 6 6 6 6 7	9
3 2 3 4 1 2 3	-1
2 65536 65537 1 2	32

Problem D. Digit DP

Input file: *standard input*
Output file: *standard output*
Time limit: 5 seconds
Memory limit: 1024 mebibytes

There are 2^n machines in a factory, numbered from 0 to $2^n - 1$. The i-th machine consumes p_i units of power. The factory uses a system called Digit Dynamic Powering (Digit DP) to control the power the machines consume.

Initially, an array $a_0, \ldots, a_{n-1}$ is given. Then the system would set the initial p_i to $\sum_{j \in S_i} a_j$, where S_i is the set of 1 bits in the binary representation of i.

After that, there may be some modifications, each modification would be adding a certain value to the power of some machines that form an interval. Formally speaking, you would be given three integers ℓ, r, x, meaning that the power consumed by each of the machines numbered between ℓ and r (inclusively) should increase by x. The endpoints of the intervals would be given as n-digit binary strings.

When some three distinct machines are used to produce a **product**, the **product**'s price should always be the product of the p_i's of those machines.

During these modifications, the manager may ask some questions about some intervals. What is the sum of the prices if we try every possible combination of three distinct machines in the interval to produce a **product**?

Formally speaking, you would be given two integers ℓ, r, meaning that you should report the sum of the products $p_i \cdot p_j \cdot p_k$ of all triples (i, j, k) satisfying $\ell \le i < j < k \le r$. As the answer may be rather large, find it modulo 998 244 353. The endpoints of the intervals would also be given as n-digit binary strings.

Input

The first line contains two integers n and q ($1 \le n \le 100$; $1 \le q \le 5 \cdot 10^4$).

The second line contains the integer array $a_0, a_1, \ldots, a_{n-1}$ ($0 \le a_i \le 10^9$).

The next q lines contain queries. On each line, the first integer t indicates the type of the query.

If $t = 1$, three integers ℓ, r, x follow ($0 \le \ell \le r < 2^n$, $0 \le x \le 10^9$).

If $t = 2$, two integers ℓ, r follow ($0 \le \ell \le r < 2^n$).

Note that ℓ and r are given in n-bit binary string format, and the leftmost bit is the highest bit.

Output

For each query of type 2, print a line with a single integer: the answer modulo 998 244 353.

Examples

standard input	*standard output*
3 3 1 2 4 2 000 111 1 010 101 1 2 000 111	1960 3040
2 2 1 1 2 00 10 2 00 11	0 2

Problem E. Except One

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

You are given three integers p, k, t, where p is a prime number.

The set S is defined as follows: $S = \{x \mid x \in N,\ 1 \le x \le p-1,\ x \ne k\}$.

Find the sum of the products of all t-element subsets of S, modulo p.

Input

The first line contains three integers p, k, t (p is a prime number; $1 \le k \le p-1$; $1 \le t \le p-2$; $p \le 10^9$).

Output

Print the sum of the products of all t-element subsets of S, modulo p.

Examples

standard input	standard output
7 5 3	1
11 6 7	3

Problem F. Fun on Tree

Input file:	*standard input*
Output file:	*standard output*
Time limit:	7 seconds
Memory limit:	256 mebibytes

Beitou, renowned for its breathtaking beauty and abundant hot springs, attracts countless tourists seeking relaxation and rejuvenation. The region boasts an extraordinary concentration of hot springs and spas, making it a global hotspot for such indulgence. Once a quaint locale where locals sought solace in its natural hot springs, the Beitou Valley has blossomed into a sprawling expanse, now housing over thirty luxurious resorts.

Traditionally, hot springs evoke imagery of volcanic activity and the unmistakable scent of sulfur. However, for some, the latter proves rather unpleasant. You find yourself among those sensitive to the smell of sulfur, which brings us to your current predicament.

Presently visiting Beitou, you've obtained a map of this enchanting place, revealing a conceptual representation as a rooted tree. It's worth noting that Beitou's mystical aura allows for unconventional measurements, even enabling negative distances between nodes. Additionally, each node on the map comes with an associated sulfur content value.

An unexpected revelation leaves you questioning the accuracy of the sulfur content assigned to each location. In order to rectify this, your mission is to align the sulfur content with the newly acquired information. Moreover, you've uncovered intel about the location of the largest volcano in the vicinity. Prioritizing your safety and well-being, your ultimate goal is to pinpoint the spot farthest from this prominent volcano while having the lowest sulfur content. Your chosen evaluation metric for each node's viability is given by the formula $d_i - a_i$, wherein d_i represents the distance between node i and the largest volcano, and a_i is the corresponding sulfur content value.

We can describe each piece of new information you acquire with three integers, x_i, y_i, and v_i. It means that all the subtree of y_i now has v_i more sulfur (if v_i is negative, then the sulfur content value decreases). Besides, you also learn that the largest volcano is actually located at x_i.

Note that the modifications of sulfur value are persistent: when you get another piece of new information, the sulfur modifications from the previous ones still apply.

Input

The first line contains two integers n and q ($2 \le n \le 2 \cdot 10^5$ and $0 \le q \le 2 \cdot 10^5$): the size of the tree and the number of queries.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($|a_i| \le 10^9$): the sulfur content value of each nodes.

Next, $n-1$ lines are given. The i-th of these lines contains two integers p_{i+1} and w_{i+1} ($1 \le p_{i+1} \le i$ and $|w_{i+1}| \le 10^9$): the parent node of vertex $i+1$ and the distance between p_{i+1} and $i+1$.

Finally, q lines are given. Each line contains three integers x_i, y_i, and v_i ($1 \le x_i, y_i \le n$ and $|v_i| \le 10^9$): the new piece of information you acquired.

Output

For each new piece of information, print a line with two integers s_i and d_i: the index of the node having the largest viability and the viability itself.

If there are multiple answers, please output the one with the minimal node index.

Examples

standard input	*standard output*
6 6 1 1 4 5 1 4 1 5 2 0 3 2 4 1 5 6 3 2 -100000 1 2 100000 1 1 0 2 2 66 3 1 5 4 4 -3	6 100005 6 10 6 10 1 4 1 -1 1 1
5 6 -10 0 2 -4 8 1 7 1 1 2 2 2 -2 1 1 100 2 1 -100 1 1 0 4 3 10 2 5 3 5 2 2	4 -87 1 17 4 13 1 19 1 17 1 15
6 3 0 0 0 0 0 0 1 10 1 10 1 -100 4 10 4 11 1 1 0 4 1 0 1 4 1000	2 10 6 11 2 10

Problem G. Game

Input file: *standard input*
Output file: *standard output*
Time limit: 25 seconds
Memory limit: 512 mebibytes

A new game called *Boundle* works as follows. In each turn, the player tells the host a string of length 5 consisting of uppercase English alphabets. Then, the host should respond with a string of length 5 consisting of `<`, `=`, `>`, indicating the alphabetical order compared to the **answer string in the host's mind** in each position. The answer string should be the same throughout the game, and should also consist of uppercase English alphabets.

Two sisters, Yui and Ui are going to play a game of *Boundle*. The elder sister Yui will be the host, and the younger sister Ui will be the player. Ui has a set of n strings she would use as queries, and for each of the n strings in Ui's set, Yui has already set a fixed response. Here's the twist though: as Yui is careless, it is guaranteed that **there actually does not exist a string** that matches all of her fixed responses.

Every turn, Ui randomly and uniformly chooses a string in her set that has not yet been chosen, and receives the fixed response for that string. For each i from 1 to n, calculate the number of ways the game is played if Ui realizes just after the i-th turn that the responses given by her elder sister Yui are not consistent. Ui is smart enough, so, when the responses don't add up, she immediately sees that **Yui does not actually have an answer string** in her mind.

Note that it is guaranteed that if Ui finishes all the strings in the set, the results she gets will not be consistent.

Also, for some mysterious reasons, unlike the original game, even if the responses have already narrowed the possible answer strings down to one possibility, the game would still continue until all the strings in the set have been chosen, or the responses become inconsistent.

Input

The first line contains an integer n ($1 \le n \le 10^5$).

Subsequently, there are n lines, with each line containing two strings. The first string belongs to the set of strings from Ui, and the second string represents Yui's response to it.

All strings in Ui's set are distinct and consist of five uppercase English alphabets. Yui's responses are strings of length 5, containing only the characters "<", "=", and ">".

It is guaranteed that there does not exist a string that matches all of Yui's responses.

Output

Print a line with n integers, the i-th of which should be the number of ways the game can be played if Ui realizes Yui is lying just after the i-th turn.

Print all numbers modulo 998 244 353.

Examples

standard input	*standard output*
2 FAKER >>>>> CHOVY =====	0 2
3 BVHUQ ><>>< YJCEQ <<><< SXXWZ >>==>	1 4 0
8 IFSXA >><<= ZAKDA <>=>= UZVAA <<<>= MTACA <>>>= RJKVA <><<= IOXOA >=<<= MRMHA ><<<= BYFWA ==<>=	0 16 108 396 816 720 0 0
8 BRKPR ><=<> VUCTO <<=<= PTCDB <<=>> PHMGV <><>< FGWHD >><>> SUSFH <<<<> IOLDD <<<<> WJPXX <><<<	0 14 120 444 744 360 0 0

Problem H. Harumachi Kaze

Input file:	*standard input*
Output file:	*standard output*
Time limit:	30 seconds
Memory limit:	256 mebibytes

This is an interactive problem.

You are given two arrays a and b of length n, consisting of non-negative integers.

There is a hidden permutation of integers from 0 to $2^{64}-1$: $p(0), p(1), \ldots, p(2^{64}-1)$. You only know that $p(0) = 0$.

Since p is a permutation, p^{-1} can also be defined: $p^{-1}(x) = y$ when $p(y) = x$.

Also, you are given an integer B. A positive integer x is called **cute** if and only if the two following conditions hold:

- The binary representation of x, when viewed as a string of length B, is a palindrome.
- If a bit in the binary representation of x is 1, this bit must either be in the first 6 bits or in the last 6 bits. For example, if $B = 14$, the two bits in the center must both be 0.

You have to support queries of two types.

The first type of query is to change an element in one of the arrays.

The second type of query is to answer the following question: if we list $2n$ integers, $p(a_1)$, $p(a_1)+p(a_2)$, $\ldots$, $p(a_1)+p(a_2)+\ldots+p(a_n)$ and $p(b_1)$, $p(b_1)+p(b_2)$, $\ldots$, $p(b_1)+p(b_2)+\ldots+p(b_n)$, and sort them into $c_1, c_2, \ldots, c_{2n}$, what would $p^{-1}(c_k)$ be? It is guaranteed that k is a **cute** number.

There are two interaction functions for you to call.

- $\text{add}(x, y)$: returns $p^{-1}(p(x) + p(y))$.
- $\text{cmp}(x, y)$: returns $p^{-1}(\min(p(x), p(y)))$.

Please beware that it is invalid to ask $\text{add}(x, y)$ if $p(x) + p(y) \geq 2^{64}$.

To help you refrain from making invalid calls, it is guaranteed that, at any moment, the following condition holds: $\max(p(a_1)+p(a_2)+\ldots+p(a_n), p(b_1)+p(b_2)+\ldots+p(b_n)) < 2^{64}$.

Input

You begin the interaction by reading three integers: n, q, B ($1 \leq n \leq 1.6 \cdot 10^4$; $1 \leq q \leq 2 \cdot 10^4$; $1 \leq B \leq 16$).

Then, you should read two lines, the first containing the array $a_1, a_2, \ldots, a_n$ ($0 \leq a_i < 2^{64}$), and the second containing the array $b_1, b_2, \ldots, b_n$ ($0 \leq b_i < 2^{64}$).

After that, you should read the contents of the q queries.

For the next q lines, the first integer *type* indicates the type of query ($1 \leq \textit{type} \leq 2$).

- If $\textit{type} = 1$, three integers follow: t, pos, x ($1 \leq t \leq 2$; $1 \leq pos \leq n$; $0 \leq x < 2^{64}$).
  - If $t = 1$, set $a_{pos} = x$.
  - If $t = 2$, set $b_{pos} = x$.
- If $\textit{type} = 2$, one integer k follows ($1 \leq k \leq \min(2^B - 1, 2 \cdot n)$). It is guaranteed that k is a **cute** number.

It is guaranteed that there is at least 1 and at most 5000 queries of type 1.

Interaction Protocol

After reading all the input, you can make at most $1.6 \cdot 10^6$ function calls.

For each call, you print a line of the form "t x y", where t is either "`A`" or "`C`", and x and y are integers ($0 \le x, y < 2^{64}$). Then flush the output, and read a line with the resulting integer z:

- If t is "`A`", you will get $z = p^{-1}(p(x) + p(y))$.
- If t is "`C`", you will get $z = p^{-1}(\min(p(x), p(y)))$.

If you make too many calls or make an invalid call, you will receive the `Wrong Answer` verdict.

After you have calculated the answers to all queries of type 2, print two lines. The first line should be "`!` m", where m is the number of type 2 queries. The second line should contain m integers $q_1, \ldots, q_m$: the answers to the type 2 queries respectively. Note that printing the answer does not count towards your total of $1.6 \cdot 10^6$ calls. After printing these two lines, your program should terminate.

Example

standard input	*standard output*
2 3 2 1 3 5 7 2 3 1 2 2 9 2 3	
	A 1 3
4	
	A 5 7
12	
	C 4 12
12	
	A 5 9
14	
	C 4 14
4	
	! 2 12 4

Problem I. Interval Addition

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	512 mebibytes

You are given an array a of n integers. You can perform operations on this array. In a single operation, you can add any real number x to some consecutive interval of a.

Determine the minimum number of operations that have to be performed to make all elements of a equal to 0.

Input

The first line contains an integer n ($1 \le n \le 23$).

The second line contains the array $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$).

Output

Print a line with a single integer: the minimum number of operations needed.

Examples

standard input	*standard output*
5 1 2 3 2 1	3
6 1 1 4 5 1 4	4

Problem J. Joining Cats

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

Suzukaze Aoba has a magical fan. There are n cats sitting on a straight line. Aoba wonders if she can merge all cats into one by using the magical fan k times.

The i-th activation of the fan produces a gust of wind of strength exactly s_i.

For each gust of wind, Aoba picks its starting position and direction (eastward or westward). Then the gust of wind starts moving continuously in that direction with constant speed. Aoba can decide to turn the fan off at any moment, in which case, the wind disappears.

When the gust of wind encounters a cat, if the weight of the cat is strictly larger than the strength of the wind, the fan turns off automatically. Otherwise, the cat will be continuously pushed to the direction the wind is blowing.

When two cats meet, they merge into a cat whose weight is equal to the sum of their weights. The above rules then apply to the newly merged cat.

Determine if joining all cats is possible by using the fan at most k times.

Input

The first line contains two integers n and k ($1 \le n, k \le 5000$).

The second line contains n integers $w_1, \ldots, w_n$ ($1 \le w_i \le 10^9$), denoting the initial weights of the cats from left to right. No two cats initially sit on the same spot.

The third line contains k integers $s_1, \ldots, s_k$ ($1 \le s_i \le 10^9$).

Output

Print a line with one word (case-sensitive): "`Yes`" if it is possible to merge all cats into one by using the magical fan at most k times, or "`No`" otherwise.

Examples

standard input	*standard output*
5 2 1 1 1 1 1 2 2	Yes
6 7 3 2 1 1 2 3 2 2 2 2 2 2 2	No
7 4 1 2 3 4 3 2 1 3 3 3 3	Yes
5 1 5 4 3 2 1 10	Yes

Problem K. Keychain

Input file: *standard input*
Output file: *standard output*
Time limit: 10 seconds
Memory limit: 256 mebibytes

Consider a two-dimensional plane and n points $p_1, \ldots, p_n$ on it. Consider n circles $C_1, C_2, \ldots, C_n$: the i-th circle is centered at p_i. All the radii of the n circles are R.

Determine the minimum value of R such that one can draw another *generalized circle* Γ that intersects all the n circles. Please find one such Γ as well.

- A circle C with radius r contains all points such that the Euclidean distance between the point and the center of the circle is exactly r.
- A *generalized circle* is either a circle or a straight line.
- We say two objects A and B intersect if they share a common point.

Input

The first line contains an integer n ($1 \le n \le 3000$). On each of the next n lines, there will be two integers x_i and y_i indicating the coordinates of point p_i ($0 \le x_i, y_i \le 10^5$). It is guaranteed that no two given points coincide.

Output

On the first line, print the optimal answer R_{opt}.

Your output should satisfy $0 \le R_{opt} \le 10^5$.

It can be proved that the minimum value exists and is in this range.

Suppose that Γ_{opt} intersects all $C_1, \ldots, C_n$ when $R = R_{opt}$.

It can be shown that, under the constraints in this problem, Γ_{opt} can be chosen to be either a circle centered at a rational coordinate, or a straight line with integer coefficients.

- In the circle case, print "C X Y Z r", which means that the radius is r, and the center of the circle is $O = (X/Z, Y/Z)$.

  The values X, Y, Z must be integers with absolute value not greater than 10^{18}. The value r should be a non-negative real number not greater than 10^{18}.
- In the straight line case, print "L a b c", which means that the line L satisfies the equation $ax + by = c$.

  The values a, b, c must be integers with absolute value not greater than 10^{18}.

When checking your answer, the jury will first check whether Γ_{opt} intersects each of the C's. This will be judged by checking:

- if $|R - r| - \varepsilon \le d(O, p_i) \le R + r + \varepsilon$ in the circle case ($d(O, p_i)$ is the Euclidean distance between p_i and O),
- or $R \le d(L, p_i) + \varepsilon$ in the line case ($d(L, p_i)$ is the distance from point p_i to line L).

Here, $\varepsilon = 10^{-6}$.

After that, your answer will be considered correct if the absolute or relative error between your R_{opt} and jury's R_{opt} doesn't exceed 10^{-6}.

Examples

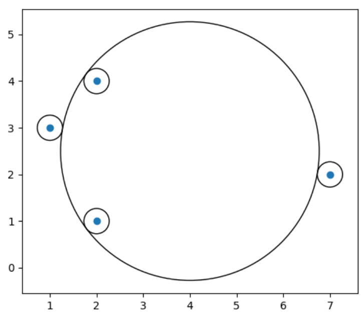
standard input
4 2 1 1 3 2 4 7 2
standard output
0.27069063257455492223 C 1152 720 288 2.77069063257455492234

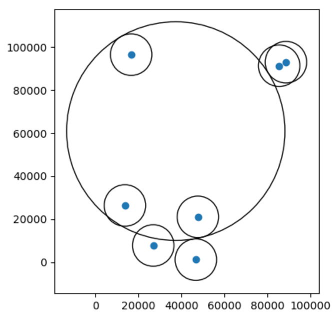
standard input
7 26919 7739 85584 91359 47712 21058 13729 26355 16636 96528 88747 93023 46770 1150
standard output
9663.87959749101919015857 C 3605577680770432 5873755742321056 96368792608 50864.33205303458045065668

standard input
10 756 624 252 208 504 416 378 312 203 287 329 391 0 0 707 703 126 104 581 599
standard output
46.05915288207108030175 L -1248 1512 90300

Note

The first two examples:





Be careful of overflow. Consider using `long double` or `__int128`.

Problem L. Lines

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

Given are three arrays of $n+1$ integers: a, b, c.

We define $3n+1$ functions $F_0, F_1, \ldots, F_{3n}$ as follows:

$$F_i(t) = it + \max_{\substack{0 \le x,y,z \le n \\ x+y+z=i}} (a_x + b_y + c_z).$$

A function F_i is said to be **NeVeR_LosEs** if and only if there does not exist a real number t such that $F_i(t) > F_j(t)$ for all $j \neq i$.

Your task is to find out which functions can be called **NeVeR_LosEs**.

Input

The first line contains an integer n ($1 \le n \le 3 \cdot 10^5$).

The second line contains the array $a_0, a_1, \ldots, a_n$ ($0 \le a_i \le 10^9$).

The third line contains the array $b_0, b_1, \ldots, b_n$ ($0 \le b_i \le 10^9$).

The fourth line contains the array $c_0, c_1, \ldots, c_n$ ($0 \le c_i \le 10^9$).

Output

On the first line, print an integer m, the number of functions that can be called **NeVeR_LosEs**.

On the second line, print m integers $0 \le i_1 \le \ldots \le i_m \le 3n$, the indices of these functions in ascending order.

Examples

standard input	*standard output*
3 3 1 8 7 9 1 3 1 5 1 1 6	5 1 3 4 7 8
1 1 2 1 2 1 2	2 1 2