Problem A. Advanced Monopoly

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

One day, Little W and Little H were playing Advanced Monopoly. This is a version of Monopoly with unique rules. The game map is a rooted tree with n vertices, where vertex 1 is the root. Each vertex on the tree has a price, denoted as w_x for vertex x.

For two distinct vertices x and y on the tree, if x is an ancestor of y (in other words, x lies on the simple path from vertex 1 to vertex y), then x is said to *dominate* y.

During the game, Little W and Little H take turns to purchase an unpurchased vertex on the tree until all n vertices are purchased by either Little W or Little H. At the start of the game, none of the vertices have been purchased.

For a purchase action, suppose the buyer purchases vertex x. They must first pay w_x game coins to the system. At this point, if x dominates n_1 vertices that have already been purchased by the opponent, and is dominated by n_2 vertices that have already been purchased by the opponent, then:

- If $n_1 > n_2$, the opponent must pay $n_1 - n_2$ game coins to the buyer.
- If $n_1 < n_2$, the buyer must pay $n_2 - n_1$ game coins to the opponent.

Both Little W and Little H are extremely smart and will adopt optimal strategies to maximize their profit in the game. Now, Little W wants to ask you: if he moves first, how many game coins can he ultimately earn? In other words, what is the total number of game coins Little W receives from Little H minus the coins he pays to the system and to Little H during the entire game? You may assume that both Little H and Little W have sufficient game coins at the start. Note that the answer could be negative.

Input

The first line of the input contains an integer n, the number of vertices ($2 \le n \le 2 \cdot 10^5$).

The second line contains n integers $w_1, w_2, \ldots, w_n$, where w_i is the price of vertex i ($0 \le w_i \le 2 \cdot 10^5$).

The third line contains $n-1$ integers $p_2, p_3, \ldots, p_n$, where p_i represents the parent of vertex i ($1 \le p_i \le n$). You may assume that the given graph is a rooted tree with vertex 1 as the root.

Output

Print one integer, the answer to the problem.

Examples

standard input	*standard output*
7 0 0 1 0 0 0 0 1 1 2 2 3 3	2
9 4 2 0 1 3 0 2 1 1 1 9 3 2 2 2 9 2	-7

Problem B. Big Staircase

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

Scrooge McDuck bought a castle at a sale and wants to restore it.

He is going to start with the main staircase. The staircase consists of n steps (numbered from 1 to n), but many of the steps are damaged.

Some are missing fragments and are therefore lower than they should be, while stones have fallen onto others, making them higher than they should be.

There are k steps that have been preserved in perfect condition and have beautiful carved reliefs. Scrooge wants to keep all of these steps untouched, and repair some of the remaining steps so that the height of step i is strictly less than the height of step $i + 1$ for every i from 1 to n. When repairing a step, the workers may increase or decrease its height by any amount. The final height of a step must be an integer and non-negative.

Repairing each step costs money, and Scrooge is extremely stingy, so he wants to solve the problem by restoring as few steps as possible.

You are given a list of n numbers representing the current heights of the steps, as well as the positions of the k steps that must be preserved. Your task — to propose a way to restore the staircase that requires restoring the minimum possible number of steps, or determine that restoring the staircase is impossible.

Input

The first line of the input contains two integers n and k — the number of steps and the number of steps that must be preserved, respectively ($2 \le n \le 10^5$, $0 \le k \le n$).

The second line contains n integers s_i — the heights of the steps ($0 \le s_i \le 10^9$). The i-th of these numbers gives the height of the i-th step.

The third line contains k pairwise distinct integers t_i — the indices of the steps that must be preserved ($1 \le t_i \le n$).

Output

If it is impossible to obtain a strictly increasing staircase under any modification of the steps that does not affect the steps that must be preserved, output -1. Otherwise output n integers — the heights of the steps of the restored staircase. The heights of the k steps whose indices are listed in the input must remain unchanged. The number of restored steps must be minimal.

Example

standard input	standard output
10 2 5 4 6 5 9 17 18 19 20 4 3 7	0 4 6 7 9 17 18 19 20 21

Problem C. Clockwork Concert in D

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

In the old clock tower, Clara prepares a concert for a choir of clockwork chimes. Each chime rings periodically, and the i-th chime rings once every r_i ticks.

For the final rehearsal, Clara must merge exactly two chimes into one deeper chime. You are given the n current periods $r_1, r_2, \ldots, r_n$. Clara needs to perform the following steps exactly once:

- Choose integers i and j such that $1 \le i < j \le n$.
- Add a new chime whose period is $(r_i + r_j)$ to the choir.
- Remove the two old chimes with periods r_i and r_j from the choir.

There are a total of $\frac{n(n-1)}{2}$ possible choices, each resulting in a choir of $n-1$ chimes. For each of these choirs, calculate the length of the first full moment when all its chimes ring together again, that is, the least common multiple of all periods in it. Then find the sum of these least common multiples modulo 998 244 353.

Input

The first line of input contains an integer n, the number of chimes ($2 \le n \le 5 \cdot 10^5$).

The next line contains n positive integers $r_1, r_2, \ldots, r_n$: the periods of the chimes ($1 \le r_i \le 10^6$).

Output

Output a single integer: the sum of least common multiples of all resulting choirs modulo 998 244 353.

Example

standard input	*standard output*
3 2 3 4	40

Note

In the example:

- When $i = 1$ and $j = 2$, the resulting periods are $\{4, 5\}$, and the least common multiple is 20;
- When $i = 1$ and $j = 3$, the resulting periods are $\{3, 6\}$, and the least common multiple is 6;
- When $i = 2$ and $j = 3$, the resulting periods are $\{2, 7\}$, and the least common multiple is 14.

Problem D. Draft Lottery

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

In the Northern Basketball Association (NBA), players are distributed among teams through a draft. The idea of the draft is that the teams that performed worst in the previous season choose new players first, thus getting a chance to acquire more promising basketball players and, as a result, build a strong roster in the future. To avoid obvious "tanking", among the 14 teams that did not reach the playoff round, a lottery is held for the right to make the first pick.

A certain number of balls with team names inside are placed into an urn for the drawing. The share of balls with the name of the team from the i-th place from the bottom is equal to a_i percent, while the sequence a_i is non-increasing and its sum is 100. The values of the numbers a_i and the further selection procedure are described in the Collective Bargaining Agreement (the contract between the teams and the league that defines the NBA regulations).

You are tasked with preparing the draft lottery. You have read the section of the Collective Bargaining Agreement that defines the values of the numbers a_i, and now you want to choose the smallest possible number of balls so that the team names can be distributed according to the required ratio.

Input

The input contains 14 real numbers a_i in non-increasing order — the percentage of balls with the name of the team that took the i-th place from the bottom ($0 < a_i \leq 100$, the sum of all a_i is 100, and all a_i are given with exactly one digit after the decimal point).

Output

Output one integer — the smallest number of balls so that the team names can be written on them in the required ratio.

Example

standard input	standard output
14.0 14.0 14.0 12.5 10.5 9.0 7.5 6.0 4.5 3.0 2.0 1.5 1.0 0.5	200

Problem E. Easy Game For Two Novices

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

Little I and Little J are playing a card game called "Fire Train", also known as "Trolley" and "Little Cat Fishing". The rules of the game are as follows (note that they may differ from the usual rules if you are familiar with the game):

- There are $2n$ cards, where for integers $1 \leq i \leq n$, there are exactly 2 cards with the face value of i.
- At the start of the game, Little I and Little J each take n cards to form their initial hands.
- A common card pile is maintained (which can be viewed as a stack), initially empty. Little I and Little J take turns, with Little I going first. During each turn, the active player performs the following operations in order:
  1. Place one card from the player's hand on top of the common card pile;
  2. If there are two identical cards in the common card pile at this moment, those two identical cards and all cards between them are moved from the common card pile to the current active player's hand;
  3. If the current active player has no cards left in hand at this moment, they lose, and the other player wins.

Little J is a novice at cards, so he will act according to the following strategy:

- Maintain a queue, initially placing the n cards in hand in a certain order into the queue;
- Each turn, place the card at the front of the queue on top of the common card pile;
- If Little J places a card and there are now two identical cards in the common card pile, the cards obtained are added to the end of the queue in the order they appear from top to bottom in the common card pile.

Little I has learned about Little J's strategy and the order of cards in the queue by peeking. Now, Little I not only wants to win but also wants to do so quickly, using the minimum number of actions to win, but he is also a novice at cards.

Therefore, given the n cards in Little J's queue and their order, you need to provide Little I's strategy so that he can win with the least number of actions, or tell him that it is impossible.

Input

The first line of the input contains an integer n indicating the number of distinct card values ($1 \leq n \leq 3 \cdot 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$, describing the cards in Little J's queue from the front to the back in order ($1 \leq a_i \leq n$, each integer appears at most twice in the sequence a).

Since Little J's n cards define what Little I's n cards will be, Little I's hand is not mentioned in the input.

Output

If Little I cannot win, simply output the integer -1; otherwise, output an integer s on the first line, indicating the number of actions Little I takes according to your strategy. On the next line, output s integers, describing the face values of the cards that Little I places on the common card pile during each action, separated by spaces. Note that the strategy you provide must ensure that s is minimized.

Examples

standard input	*standard output*
3 1 3 3	3 2 1 2
1 1	-1

Problem F. Find The Number of Stacks

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

In a galaxy far, far away, a new service has spread — starship rental, or spacesharing.

All rental starships are standardized and have the same mass and shape, as close to cubic as possible. This makes it possible to park starships in hangars by stacking them on top of each other in towers, saving scarce space.

However, since the starships were manufactured by different companies, they differ in at least one parameter: hull strength. If a starship's hull has strength x, then at most x starships can be placed on top of it. That is, for example, if we have starships with strengths 0, 1, and 3, then we can assemble them into a single tower in exactly one way — placing the starship with strength 3 at the bottom, then the starship with strength 1 on top of it, and then the starship with strength 0 on top. If we have four starships of strength 2, then one tower is not enough (the load on the bottom starship in the tower would be too large), and we will have to make two towers.

The corporation plans to expand the service to yet another planet. The rental fleet has already been allocated; all that remains is to rent a hangar. Given the list of starships and their strengths, determine the minimum number of towers into which they can be "packed".

Input

The first line of the input contains one integer n — the number of starships ($1 \le n \le 10^5$).

The second line contains n integers x_i — the strengths of the starships ($0 \le x_i \le 10^5$).

Output

Print one integer — the minimum number of "towers" into which all n starships can be assembled.

Examples

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

Problem G. Governor Scroogius

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

Scroogius, the governor of some far province in the ancient Rome received an urgent request from local authorities for funding intelligence activities to prevent the rebellion. The request was formatted as a long string of Roman numerals, without spaces.

Governor then wrote the request to Caesar, having previously placed spaces in the request so that the string represented a sequence of correctly written Roman numbers with the maximum possible sum. After receiving the requested money, the governor arranged the spaces in the original request string so that the string represented a sequence of correctly written Roman numbers with the minimum possible sum (the number of spaces in the first and second arrangements could differ), and sent this amount of money to the local authorities along with the new decryption of the string. Legend says that Scroogius profit from this venture is exactly n.

Unluckly, the legend does not tell the exact string of the Roman numerals, but tells only value of n. Check if such a situation is possible, and if so, what could the shortest string look like from which such a story could have occurred.

Your task is to answer this question for a given value of n. Moreover, the string you must output should be lexicographically smallest.

We remind you of the way Roman numbers are written (the table is taken from Wikipedia):

Place Value	Thousands	Hundreds	Tens	Units
1	M	C	X	I
2	MM	CC	XX	II
3	MMM	CCC	XXX	III
4		CD	XL	IV
5		D	L	V
6		DC	LX	VI
7		DCC	LXX	VII
8		DCCC	LXXX	VIII
9		CM	XC	IX

Note that:

- The numbers 4, 9, 40, 90, 400, and 900 are written in reverse notation, where the first symbol is subtracted from the second (for example, for 40 (`XL`), '`X`' (10) is subtracted from '`L`' (50)). This is the **only** place where reverse notation is used.
- A number containing multiple decimal digits is constructed by appending the Roman equivalent of each digit from the most significant to the least significant.
- If the decimal place is 0, no digits are written in that place in the Roman representation.
- The largest number that can be represented in the Roman numeral system is 3,999 (MMMCMXCIX).

Input

The first line of the input contains a single integer n — Governor's profit ($0 \le n \le 10^5$).

Output

If it is impossible to achieve the difference of n for any string, output -1. Otherwise, output the shortest string composed of the letters `I`, `V`, `X`, `L`, `C`, `D`, and `M`, for which the difference between partitions is exactly n. If there are multiple such strings, output the lexicographically smallest one.

Examples

standard input	standard output
0	C
1	-1

Problem H. Hard? NP-Hard!

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

In this problem, you need to solve the complete knapsack problem.

There are n types of items: an item of the i-th type has a volume of v_i and a value of c_i.

There are q queries, each providing the capacity x of the knapsack. You need to select several items, with each type of item being selectable in any quantity (including not selecting it at all), in order to maximize the total value of the selected items, under the condition that the sum of the volumes of the selected items is exactly x. You need to provide this maximum total value or report that there is no solution where the total volume is exactly x.

To make you demonstrate your ability to solve an NP-Hard problem, x will be much larger than v_i, as detailed in the input constraints.

Input

The first line of the input contains two integers, n and q, representing the number of item types and the number of queries, respectively ($2 \le n \le 50$, $2 \le q \le 10^5$).

Each of the next n lines contains two integers, v_i and c_i, describing one type of item ($1 \le v_i \le 10^5$, $1 \le c_i \le 10^6$).

Each of the following q lines contains a single integer x describing the capacity of the knapsack in a query ($10^{11} \le x \le 10^{12}$).

Output

For each query, output a single line with an integer. If there is no solution where the total volume is exactly x, output -1; otherwise, output the maximum total value of the selected items.

Example

standard input	*standard output*
2 2 10 17 14 25 100000000003 100000000004	-1 178571428576

Note

Note

For the first query, every item volume is even, so every achievable total volume is even. Therefore, there is no way to obtain the capacity 100000000003 exactly.

For the second query, the optimal solution is to select 3 items of type 1 and 7142857141 items of type 2:

$$3 \cdot 10 + 7142857141 \cdot 14 = 100000000004,$$

and the corresponding maximum total value is

$$3 \cdot 17 + 7142857141 \cdot 25 = 178571428576.$$

Problem I. Invisible Jewels and Judge J

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

This game is called "The Sealed Box Game". The rules of the game are as follows. One player acts as the illusionist, and the other player acts as the judge. The illusionist can secretly place x invisible jewels, where x is an integer in the range $[0, n]$, into a sealed box, while the judge needs to guess the amount in the box. Suppose the judge guesses y, where y is also an integer in the range $[0, n]$. If $x = y$, the trick is exposed, and the illusionist receives nothing. If $x > y$, the trick succeeds, and the illusionist receives x prize tokens from the judge. If $x < y$, the trick fails, but since the judge was still wrong, they must pay the illusionist $y/2$ prize tokens (this amount is not necessarily an integer). The game is played for a limited number of rounds, with both players taking turns as the illusionist and the judge.

It can be proven that, if both parties play optimally, the illusionist will adopt the same probabilistic strategy in each round, and the judge will also adopt the same probabilistic strategy in each round. Your task is to output the optimal strategies for both sides in a round.

Input

The first line of the input contains an integer n ($1 \le n \le 4 \cdot 10^5$).

Output

Output two lines, each containing $n + 1$ integers, where the i-th integer represents the probability of hiding or guessing $i - 1$ jewels, written modulo 998 244 353 (in other words, if the probability is p/q, print $p \cdot q^{-1} \bmod 998\,244\,353$).

The first line outputs the illusionist's strategy, and the second line outputs the judge's strategy.

You need to ensure that, while one side's strategy remains unchanged, the other side cannot increase their expected profit regardless of how they change their strategy.

It can be proven that such a strategy is unique.

Example

standard input	*standard output*
2	499122177 748683265 748683265 873463809 748683265 374341633

Note

The six numbers in the example are: 1/2, 1/4, 1/4, 1/8, 1/4, and 5/8.

Problem J. Juggling With Sequences

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

You have several positive integers within the range $[1, n]$: for $1 \leq i \leq n$, you have a_i integers of value i. Let $s = \sum_{i=1}^{n} a_i$.

For a sequence $p_1, p_2, \ldots, p_k$, its mode $\text{maj}(p_1, p_2, \ldots, p_k)$ is defined as the number that appears most frequently. If there are multiple numbers that appear the most frequently, the largest of those numbers is considered the mode.

Now you need to arrange these s numbers into a sequence $b_1, b_2, \ldots, b_s$ such that $\sum_{i=1}^{s} \text{maj}(b_1, b_2, \ldots, b_i)$ is maximized. Output this maximum value.

Input

The first line of the input contains a single integer n: the range of values ($2 \leq n \leq 10^5$).

The next line contains n positive integers $a_1, a_2, \ldots, a_n$: the count of each number ($1 \leq a_i \leq 10^5$).

Output

Output a single line with an integer representing the maximum value of $\sum_{i=1}^{S} \text{maj}(b_1, b_2, \ldots, b_i)$.

Example

standard input	*standard output*
3 1 3 2	17

Note

One sequence that achieves the maximum value is $(3, 2, 3, 1, 2, 2)$.

Problem K. Kinky Chase Festival

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

The veteran 2D mobile game "La Lumière: Scarlet Intense Flame" will stop operations this March. As a loyal player of this game, Little S hopes to achieve a special score during the last event of the game, marking a perfect conclusion to the unforgettable times spent with this game over the past decade.

Each event in "La Lumière: Scarlet Intense Flame" has its unique rules, and the last event is the Kinky Chase Festival. In the Kinky Chase Festival, players need to repeatedly conquer randomly generated multi-layer mazes. Each time they exit the maze, the score for that random maze is calculated based on how well each layer of the maze is completed. The process for challenging the maze is simplified as follows:

1. Choose the difficulty of the maze to challenge. Little S is an experienced player of this game, so in this problem, we assume that Little S always challenges the highest difficulty maze. The highest difficulty maze has a maximum depth of n layers. After determining the difficulty, the challenge begins from layer 1 of the randomly generated maze.
2. Challenge layer i. When challenging layer i, Little S may fail the challenge, succeed normally, or succeed with high honors. If Little S chooses a conservative challenge strategy, there is a probability of $p_{i,0}$ of failing the challenge, a probability of $p_{i,1}$ of succeeding normally, and a probability of $p_{i,2}$ of succeeding with high honors; if Little S chooses an aggressive challenge strategy, there is a probability of $q_{i,0}$ of failing the challenge, a probability of $q_{i,1}$ of succeeding normally, and a probability of $q_{i,2}$ of succeeding with high honors.
   - When succeeding normally, a score of $s_{i,1}$ is earned for the current layer; when succeeding with high honors, a score of $s_{i,2}$ is earned for the current layer. This portion of the score will not be directly added to the player's total score but will be calculated upon exiting the maze. If the challenge is successful and it is not the last layer (that is, $i < n$), proceed to step 3 to choose whether to continue challenging; otherwise (if $i = n$), exit the maze and proceed to step 4 for scoring.
   - If the challenge fails, the player is forced to exit the maze and proceeds to step 4.
3. If it is not the last layer, the player can choose whether to continue challenging the next layer. If the player chooses to continue, return to step 2; otherwise, exit the current maze and proceed to step 4 for scoring.
4. Scoring for this maze: If forced to exit due to failure, no rewards are earned for the current layer, and the scores accumulated from previous layers in this maze must be multiplied by the penalty coefficient c (to ensure the final score is an integer, the game will sum the penalized scores and then take the floor); aside from forced exits, players can earn all unscored points when they voluntarily exit or exit after completing the maze.

Little S aims to achieve a relatively high target score, so Little S needs to repeatedly challenge the highest difficulty mazes and then, as the target score approaches, choose a relatively stable strategy based on the remaining score to ensure that the target score is exactly achieved by the end of the event. Little S does not know how to program, so Little S has come to you, hoping you can help calculate the maximum probability of achieving exactly the target score when the remaining score is between 1 and m, following the above process and using the optimal strategy.

Input

The first line of input contains three integers, n, m, and c', where n and m have the same meanings as in the problem statement, and $c' = 100c$ ($1 \le n \le 6$, $1 \le m \le 10^4$, $0 \le c' \le 100$).

Each of the next n lines contains eight integers: $s_{i,1}$, $s_{i,2}$, $u_{i,0}$, $u_{i,1}$, $u_{i,2}$, $v_{i,0}$, $v_{i,1}$, $v_{i,2}$, where $s_{i,1}$ and $s_{i,2}$ represent the scores corresponding to succeeding normally and succeeding with high honors during the challenge; $u_{i,j}$ and $v_{i,j}$ represent the probability weights for the conservative and aggressive challenge strategies, respectively: $p_{i,j} = \frac{u_{i,j}}{u_{i,0} + u_{i,1} + u_{i,2}}$, $q_{i,j} = \frac{v_{i,j}}{v_{i,0} + v_{i,1} + v_{i,2}}$ ($1 \le s_{i,1} \le s_{i,2} \le 10^4$, $0 \le u_{i,j}, v_{i,j} \le 10^4$, $u_{i,1} + u_{i,2} \ge 1$, $v_{i,1} + v_{i,2} \ge 1$).

Output

Output a single line containing m real numbers, where the i-th number (for $1 \le i \le m$) represents the maximum probability of achieving exactly i points when the remaining score is exactly i under the optimal strategy. Your output will be considered correct if the absolute or relative error of each real number in your output does not exceed 10^{-6}.

Example

standard input
2 8 60 2 5 0 2 1 0 1 2 3 6 1 2 1 2 1 2
standard output
0.266666666666666667 0.666666666666666667 0.355555555555555556 0.480000000000000000 0.873333333333333333 0.408888888888888889 0.804444444444444444 0.768037037037037037

Note

The example output is broken into multiple lines for readability.