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. Bonnie and the Crumb Critic

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

Bonnie runs a small bakery, and today a very strict food critic is visiting it. The critic is known as the Crumb Critic: he likes tasting pastries, but he also enjoys making bakers nervous.

Bonnie has an infinite amount of identical blank cupcakes to prepare. A cupcake is finished only after n chocolate chips have been placed on it. Each minute Bonnie can place k chocolate chips in total, distributing them among any cupcakes she chooses. There is no way to place a fraction of a chocolate chip.

The Crumb Critic has just left the counter. From now on, at regular one-minute intervals, he will come back and take away one unfinished cupcake. If there are several unfinished cupcakes, he chooses which one to take so as to delay Bonnie as much as possible. A finished cupcake is no longer unfinished and can be shown to the critic.

Bonnie wants to show a finished cupcake to the Crumb Critic as soon as possible. What is the minimum time in minutes required to do it? Assume that the Crumb Critic is determined to maximize this time.

Input

The single line of the input contains two integers, n and k ($1 \leq k \leq n \leq 10^5$).

Output

If Bonnie cannot finish a cupcake no matter what, print -1.

Otherwise, output a line with a positive integer representing the answer.

Example

standard input	*standard output*
3 2	2

Note

In the example, during the first minute, Bonnie first chooses two cupcakes and places one chocolate chip on each, then the Crumb Critic comes and takes one away. After that, Bonnie only needs to place two chocolate chips on the remaining cupcake. Therefore, when the critic comes back for the second time, Bonnie will have a finished cupcake.

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 $1 \leq i < j \leq n$.
- 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 ($2 \leq n \leq 5 \cdot 10^5$).

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

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. Delicious Apples

Input file:	*standard input*
Output file:	*standard output*
Time limit:	7 seconds
Memory limit:	1024 mebibytes

A farmer has planted an apple tree, which is laden with apples of various sizes. Summer is the prime season for the growth and development of fruit trees, as new branches continuously sprout and new apples are formed, while the existing apples continue to grow.

To observe and record the growth of the apples, in order to have a rough estimate of future harvests, the farmer has conducted long-term careful observations and studies. At the very beginning of the recording period, there are a total of n apples on the tree, numbered from 1 to n, and there are $n-1$ branches connecting these apples, with each branch connecting exactly one apple at each end, forming a tree structure. The farmer has estimated the value of each apple, with the initial value of the i-th apple being a_i, which represents the net profit the farmer would gain if he picked and sold it at that moment. Considering cost factors, a_i may be negative.

During the entire recording period, there are m noteworthy events that occurred, which can be categorized into the following types:

- `1` u v w: A new apple has sprouted in the branch connecting apples u and v. Let us say there were originally k apples on the tree; now there are $k+1$ apples, and the farmer will number the newly sprouted apple as $k+1$, with a value of w. The original branch connecting apples u and v splits into two branches: one connecting apple u and apple $k+1$, and the other connecting apple $k+1$ and apple v.
- `2` u w: A new branch and a new apple have sprouted. Let us say there were originally k apples on the tree; now there are $k+1$ apples, and the farmer will number the newly sprouted apple as $k+1$, with a value of w. The new branch connects apple u and apple $k+1$.
- `3` u v w: The values of a portion of the apples have changed. The values of all apples along the entire segment of branches connecting apples u and v (the shortest path in the tree structure connecting u and v, including u and v themselves) increase by w. Note that w may be negative, for example, due to insufficient nutrients or pests.
- `4` u v w: The farmer wants to conduct a sampling survey on the tree to study his potential profits. He defines apples with a value of at least w as "high-quality apples" and selects the entire segment of branches connecting apples u and v (as defined above) to count how many "high-quality apples" are present in that segment.

However, due to the large number of apples, the farmer cannot count them all and has to ask for your help. Note that, since the farmer cannot predict the future, you must answer the questions online.

Input

The first line of the input contains two integers, n and m ($2 \le n \le 2 \cdot 10^5$, $1 \le m \le 2 \cdot 10^5$). The second line contains n integers a_i ($-10^9 \le a_i \le 10^9$).

Each of the following $n-1$ lines contains two integers, u_i and v_i, describing all the initial branches on the tree ($1 \le u_i, v_i \le n$, $u_i \ne v_i$). You may assume that the given graph is a tree.

Each of the following m lines contains either three or four integers, describing all the events in chronological order, formatted as described in the problem statement.

To reflect the online requirement, let the answer to the latest event of type `4` be *lastans*. Initially, *lastans* $= 0$. Then all input values of u', v', w' in the events must be XORed with *lastans* to get the actual u, v, w ($-10^9 \le w \le 10^9$, the apple numbers involved at any time are valid).

All events of type 1 guarantee that the branch connecting apples u and v exists at the time of the event.

Output

For each event of type 4, output one line with an integer representing the number of "high-quality apples" in the farmer's survey.

Example

standard input	*standard output*
5 6 1 3 3 2 2 1 2 1 3 2 4 2 5 4 3 4 2 3 2 6 2 4 0 7 1 1 5 6 1 2 2 7 4 0 3 0	3 4 2

Note

In the example, after removing the online requirement, the data is as follows:

```
5 6
1 3 3 2 2
1 2
1 3
2 4
2 5
4 3 4 2
3 1 5 1
4 3 4 2
1 1 2 5
2 6 3
4 4 7 4
```

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. Fuel For The Lighthouse

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

At the end of the earth, a gray-white lighthouse stands on the long coastline. This area of the sea has complex currents and rugged reefs, yet it is a necessary route for many shipping lanes. Without such a tall and bright lighthouse to illuminate the path for passing ships, perhaps more unfortunate lives would be buried in the depths of the sea. To take care of this beacon, several pairs of trained watchmen take turns guarding it. The monotonous work day after day cannot afford any mistakes, and their tense nerves can only relax when the next watchman arrives.

Before the widespread use of electricity, lighthouses typically used kerosene lamps to guide passing sailors. Whenever it is Y and S's turn to take care of the lighthouse, they are also both responsible for refueling. Each time they have to bring more fuel to the lighthouse, each of them needs to carry a fuel barrel with a volume of ℓ liters. If the barrels are not filled to the top, it does not significantly affect the normal operation of the lighthouse; it just means more trips back and forth. However, if both watchmen are thinking of being lazy, the problem may not be as simple as just a few extra trips. Y and S came up with a good idea: they would fill each other's barrels.

Y and S first need to decide who will fill the kerosene first. Then the first of them fills the other's barrel. After that, the second watchman fills the first one's barrel. At this point, Y and S will compare who filled the other's barrel more. Finally, each will carry their barrel (filled by the other person) to the lamp room.

There are n containers in the lighthouse used to pour the stored kerosene into the barrels, where the i-th container has a volume of a_i. A watchman fills the other one's barrel in steps. On each step, he randomly selects one container from all the containers with equal probability, fully fills it with kerosene, and pours all of it into the other person's barrel. The watchman can end filling the other's barrel after any number of steps, including 0.

However, there is an extra rule regarding overflows. If one of them fills the other's barrel completely, but there is still kerosene left in the container, that unfortunate watchman must carry both barrels to the lamp room alone—this adds a bit of fun to their monotonous lives.

Now there is only one question left: when Y and S both adopt optimal strategies to make the other carry as much kerosene as possible, what is the probability that the amount of kerosene carried by the first watchman is not more than that of the second watchman? Clearly, if the first watchman causes an overflow, the second watchman can always play safe and enjoy the misfortune; therefore, when the first watchman overflows, he carries more than the second if both play optimally.

Input

The first line of the input contains two integers, n and ℓ, representing the number of containers and the volumes of the barrels, respectively ($1 \le n \le 2000$, $1 \le \ell \le 10^9$).

The second line contains n integers, $a_1, \ldots, a_n$, representing the volume of each container ($1 \le a_i \le 2000$).

Output

Output a real number representing the probability that the amount of kerosene carried by the first watchman is not more than that of the second watchman. Your output will be considered correct if the absolute or relative error does not exceed 10^{-6}.

Examples

standard input	*standard output*
2 4 1 3	0.562500000000000000
4 1879 6 10 14 20	0.476800000000000000

Note

More formally, in terms of game theory, the first watchman wins if both of the following conditions hold:

1. the first watchman doesn't overflow the second watchman's barrel;
2. either the second watchman overflows the first watchman's barrel or the first watchman carries less or equal kerosene than the second watchman.

The goal of each watchman is to win the game.

In the first example, it can be proven that the first watchman's strategy must be to fill the other's barrel completely, and that doing so guarantees victory. After several random selections, the probability of exactly filling the other's barrel is:

$$2\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^4 = \frac{9}{16} = 0.5625.$$

Problem G. Get Them Equal!

Input file:	*standard input*
Output file:	*standard output*
Time limit:	7 seconds
Memory limit:	1024 mebibytes

Given a sequence of length $4n$, where each number from 1 to n appears 4 times, determine whether it is possible to partition it into two equal subsequences.

The sequences a and b are equal when their lengths are equal, and when for every valid i, $a_i = b_i$.

Input

The first line contains a positive integer n ($1 \leq n \leq 5 \cdot 10^4$).

The second line contains $4n$ positive integers, representing the sequence. It is guaranteed that each number from 1 to n appears 4 times.

Output

If it is not possible to partition the sequence into two equal subsequences, output a single line "`No`".

Otherwise, output "`Yes`" on the first line. On the second line, output a string of length $4n$ consisting of zeros and ones, where the i-th position indicates which subsequence the i-th number of the original sequence belongs to. You need to ensure that the two subsequences you create are completely equal. Any valid partition will be considered correct.

Example

standard input	*standard output*
2 1 1 2 1 2 2 1 2	Yes 10110100

Note

In the sample, both subsequences are $(1, 2, 1, 2)$.

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 "Invisible Jewels and Judge J". 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.