Problem A. Writing Lab 1 Together

- 2023.10.07 02:00 Update: Strengthened testcases and rejudged solutions.
- 2023.10.07 02:00 Update: Added subtask for $n \leq 1000$.
- 2023.10.06 17:40 Update: Added sample 3.

Problem Description

In the "Introduction to Algorithms" class, there are n students, each assigned to work on Lab 1 within a specified time range from ℓ_i to r_i.

Recognizing that Lab 1 can pose challenges for students who are just starting, they may engage in discussions with their classmates.

For each student s_i, two questions arise:

- First, is there any student s_j who can be found by s_i during all of s_j's working hours?
- Second, is there a specific classmate whom s_i can find to discuss Lab 1 at any time during s_i's work period?

Input Format

The input consists of several lines.

The first line contains an integer: n, indicating the number of students in "Introduction to Algorithms."

The next n lines contain two integers each: ℓ_i and r_i, representing the work period of the i^{th} student.

Output Format

Print a line with n integers. The i^{th} integer is 1 if the answer to the first question for student s_i is true; otherwise, it is 0.

Then, print another line with n integers. The i^{th} integer is 1 if the answer to the second question for student s_i is true; otherwise, it is 0.

Constraints

- $1 \leq n \leq 200\,000$.
- $1 \leq \ell_i < r_i \leq 10^9$ for $i = 1, 2, \ldots, n$.
- All input values are integers.

Subtasks

1. (30 points) $n \leq 1000$.
2. (70 points) No additional constraints.

No.	Testdata Range	Time Limit (ms)	Memory Limit (KiB)
Samples	1-3	1000	262144
1	1-15	1000	262144
2	1-28	1000	262144

Samples

Sample Input 1

```
4
1 6
2 4
4 8
3 6
```

This sample input satisfies the constraints of all the subtasks.

Sample Output 1

```
1 0 0 0
0 1 0 1
```

Sample Input 2

```
4
1 1000000000
2 999999999
2 999999999
3 999999998
```

This sample input satisfies the constraints of all the subtasks.

Sample Output 2

```
1 1 1 0
0 1 1 1
```

Sample Input 3

```
2
1 2
1 2
```

This sample input satisfies the constraints of all the subtasks.

Sample Output 3

```
1 1
1 1
```

The two students can find each other to discuss Lab 1.

Problem B. Falling Domino

Problem Description

Shigure Ui was inspired by a YouTube video featuring the arrangement and toppling of dominoes to create intricate patterns (similar to the picture above). Intrigued by this, she decided to purchase a set of $n \times m$ dominoes to try her hand at it.

However, she soon realized that things were not as straightforward as she initially thought. Each domino in the set had a unique weight. Additionally, for a domino positioned at (i, j) ($1 \leq i \leq n$, $1 \leq j \leq m$), it would only topple under the following conditions:

- If $i > 1$ and $j > 1$: Both dominoes located at $(i-1, j)$ and $(i, j-1)$ must have greater weights than it.
- Alternatively, if $i = 1$: The domino located at $(i, j-1)$ must be heavier.
- Finally, if $j = 1$: The domino located at $(i-1, j)$ must be heavier.

Once she arranges all the dominoes, Shigure Ui plans to use Ui Beam (ういビーーーーム) to topple the domino located at $(1, 1)$. She hopes that this will trigger a chain reaction causing all the other dominoes to fall.

Currently, Shigure Ui is holding a specific domino, which is known to be the k^{th} heaviest among all the dominoes, and wants to place it on an empty board. She is curious about how many different positions she can place this domino.

Input Format

The input consists of a single line containing three integers: n, m, and k. These values represent Shigure Ui's intention to arrange dominoes in an n by m matrix, along with the rank of the domino's weight that Shigure Ui currently holds.

Output Format

Output a single integer indicating the number of different positions where Shigure Ui can place the domino on an empty board.

Constraints

- $n, m \geq 1$.
- $n \times m \leq 10^{14}$.
- $1 \leq k \leq n \times m$.
- All input values are integers.

Subtasks

1. (30 points) $n \times m \leq 10^7$.
2. (20 points) $n = m$.
3. (50 points) No additional constraints.

No.	Testdata Range	Time Limit (ms)	Memory Limit (KiB)
Samples	1-3	1000	262144
1	4-19	1000	262144
2	20-26	1000	262144
3	1-38	1000	262144

Samples

Sample Input 1

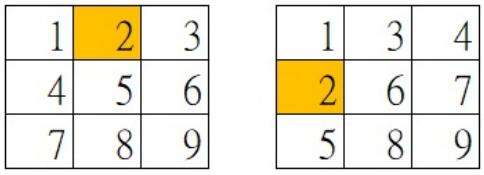
```
3 3 2
```

This sample input satisfies the constraints of all the subtasks.

Sample Output 1

```
2
```

There are two possible positions ($(1, 2)$ and $(2, 1)$) to place the second heaviest domino.



Sample Input 2

```
1 1000000 48763
```

This sample input satisfies the constraints of Subtasks 1, 3.

Sample Output 2

```
1
```

Sample Input 3

```
10000000 10000000 4878763
```

This sample input satisfies the constraints of Subtasks 2, 3.

Sample Output 3

```
75888381
```

Problem C. Convex Hull

- 2023.10.06 22:40 Update: Added sample grader interaction.

Problem Description

An upper convex hull denoted as $\mathcal{C}$ comprises $L+1$ lattice points, specifically $\{(0, y[0]), (1, y[1]), \ldots, (L, y[L])\}$.

However, $\mathcal{C}$ is not directly visible. Therefore, in order to observe $\mathcal{C}$, you need to submit queries of the following type:

- Request a slope m with an integer value, and we will provide you with the tangent point where the line $y = mx + c$ intersects $\mathcal{C}$ from above (in cases where there are multiple such points, we will furnish you with the one having the smallest x coordinate).

It is essential to note that $y[0] = y[L] = 0$, and for $i = 1, 2, \ldots, L-1$, the values of $y[i]$ are bounded within the range $1 \leq y[i] \leq 10^9$. Your objective is to determine the value of $y[k]$.

Please be aware that the convex hull may **NOT** exhibit strict convexity.

What is an upper convex hull?

An "upper convex hull" refers to the boundary that encloses the uppermost points of a set of data in a way that creates a convex shape. In a two-dimensional space, it is the smallest convex polygon that encompasses the highest points of a given dataset.

Implementation Details

You should implement the following procedure:

```
int convex_hull(int L, int k)
```

- L: the upper convex hull $\mathcal{C}$ comprises $L+1$ lattice points.
- k: your objective is to determine the value of $y[k]$.
- This procedure should return the value of $y[k]$.
- This procedure will be called at most t times.

You can call the following procedure at most 32 times per call to `convex_hull`:

```
pair<int, int> query(int m)
```

- m: the slope you request, the value must be within $[-10^9, 10^9]$.

- This procedure will return the coordinates of the leftmost point where the line $y = mx + c$ tangentially touches the upper convex hull $\mathcal{C}$ from above.

Constraints

- $1 \leq t \leq 20$.
- $2 \leq L \leq 10\,000$.
- $1 \leq k \leq L - 1$.
- $1 \leq y[i] \leq 10^9$ for $i = 1, 2, \ldots, L - 1$.
- $y[0] = y[L] = 0$.
- Points $\{(0, y[0]), (1, y[1]), \ldots, (L, y[L])\}$ form an upper convex hull.

Subtasks

1. (40 points) Any three points do not collinear.
2. (60 points) No additional constraints.

No.	Testdata Range	Time Limit (ms)	Memory Limit (KiB)
Samples	1	1000	262144
1	2-5	1000	262144
2	1-11	1000	262144

Examples

Example 1

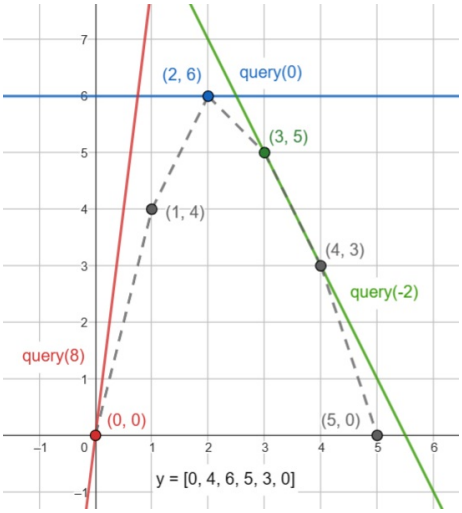
Consider a scenario with $y = [0, 4, 6, 5, 3, 0]$ and $k = 3$:

```
convex_hull(5, 3)
```

The interaction **MAY** proceeds as follows:

1. We start a query by calling `query(0)`. This query returns the point $(2, 6)$, which is shown in blue below.
2. Next, we make a query with `query(8)`, resulting in the point $(0, 0)$, illustrated by the red line.
3. Lastly, we call `query(-2)`, which gives us the point $(3, 5)$, represented by the green line.

Based on these queries, we deduce that $y[3] = 5$. Therefore, the `convex_hull` procedure returns the value 5 as the result.



The upper convex hull $\mathcal{C}_1$.

This sample input satisfies the constraints of all the subtasks.

Example 2

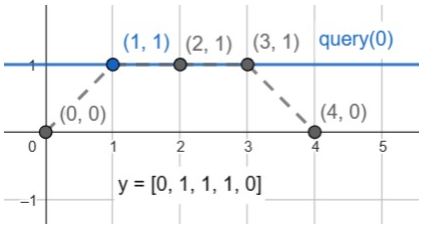
Let's consider another scenario with $y = [0, 1, 1, 1, 0]$ and $k = 2$:

```
convex_hull(4, 2)
```

Here's how the interaction **MAY** unfolds:

1. We initiate a query with `query(0)`. This query yields the point $(1, 1)$, as seen in blue below.

By observing that the highest y-value among these points is 1, and since $y[2] \neq 0$, we conclude that $y[2] = 1$. Consequently, the procedure returns the value 1 as the result.



The upper convex hull $\mathcal{C}_2$.

This sample input satisfies the constraints of Subtask 2.

Example 3

Let's consider another scenario with $y = [0, 9, 17, 21, 24, 26, 27, 0]$ and $k = 4$:

```
convex_hull(7, 4)
```

The procedure should return 24.

This sample input satisfies the constraints of all the subtasks.

Sample grader

The sample grader reads the input in the following format:

- line 1: t
- line $2+2i$ ($0 \le i \le t-1$): L k
- line $3+2i$ ($0 \le i \le t-1$): $y[0]$ $y[1]$ $\ldots$ $y[L]$

The sample grader prints your queries in the following format:

- `<k>th query: slope = <m>`: where `<k>` denotes the k^{th} call to `query`, and `<m>` denotes the parameter m you passed into `query`.

You should input the x coordinate which the function should return.

For each call to `convex_hull`, the sample grader prints `convex_hull: <y>` in the first line, where `<y>` is the return value of `convex_hull`.

If the answer is correct, the sample grader prints `Accepted: <query_count>` in the second line, where `<query_count>` is the number of calls to `query` you make.

If the answer is incorrect, the sample grader prints `Wrong Answer: <MSG>` in the second line, where `<MSG>` is one of the following:

- `wrong coordinate`: the return value of `convex_hull` is not $y[k]$.

The following describes the scenario in Example 1.

User Input	Grader Output	Notes
`1`		$t = 1$
`5 3`		First testcase: $(L, k) = (5, 3)$
`0 4 6 5 3 0`		$y = [0, 4, 6, 5, 3, 0]$
	`1st query: slope = 0`	Your program calls `query(0)`.
`2`		It should return $(2, y[2])$.
	`2nd query: slope = 8`	Your program calls `query(8)`.
`0`		It should return $(0, y[0])$.
	`3rd query: slope = -2`	Your program calls `query(-2)`.
`3`		It should return $(3, y[3])$.
	`convex_hull: 5`	Your program returns $y[k] = 5$.
	`Accepted: 3`	The answer is correct, and you have used 3 queries.

If the sample grader detects a protocol violation, the output of the sample grader is `Protocol Violation: <MSG>`, where `<MSG>` is one of the following:

- `too many queries`: `query` is called more than 32 times in any call to `convex_hull`.
- `invalid parameters`: `query` is called with m not within $[-10^9, 10^9]$.

Please note that the sample grader does **NOT** validate the input (e.g. check the convexity of $\mathcal{C}$).

Notes

- Here is a sample implementation. (Link)
- You should include `"1612.h"` in your program.
- You should **NOT** implement the `main` function.
- You should only submit `1612.cpp` to the Online Judge.
- You should **NOT** read anything from `stdin` or print anything to `stdout`.
- You can use `stderr` for debug (`std::cerr`).
- You can modify the grader as you want.
- You can use `g++ -std=c++17 -O2 -o 1612 1612.cpp grader.cpp` to compile the code, and use `./1612` or `1612.exe` to run the code.

Problem D. エロ発生 (Error)

- 2023.10.06 15:00 Update: Strengthened testcases and rejudged solutions.
- 2023.10.06 15:00 Update: Added Sample 3.

Problem Description

After three years of effort, Neko-chan successfully nurtured the rarest shiny *Noitatumrep* in the game! Regular Noitatumreps usually resemble $[n, n-1, \ldots, 1]$, but shiny Noitatumreps, on the other hand, have the exact opposite appearance, represented as $[1, 2, \ldots, n]$!

Just after Neko-chan went to bed, a sudden earthquake struck, causing a major server error in the game, resulting in the loss of a significant amount of data, known as "エロ発生." After the server maintenance was completed, Neko-chan discovered that her Noitatumrep was no longer the rare shiny.

She wanted to file a complaint with the game company, and their response was, "We will compensate you if you can prove that there is a possibility that the earthquake indeed caused the disappearance of the shiny." Neko-chan collected earthquake-related information and framed the problem in the following way:

1. Initially, Neko-chan had a shiny Noitatumrep $a = [1, 2, 3, \ldots, n-1, n]$, which turned into b ($b \neq a$) after server maintenance.
2. After that, there were q instances of "エロ発生," which caused genetic mutations in the Noitatumrep between positions $[\ell_i, r_i]$, potentially resulting in any reordering of the subarray $[a_{\ell_i}, a_{\ell_i+1}, \ldots, a_{r_i}]$.
3. Neko-chan wants to know the earliest instance of "エロ発生" after which her shiny Noitatumrep could have become b.
4. If, even after q instances of "エロ発生," it is still not possible to become b, the game company may have randomly generated a Noitatumrep for Neko-chan. In this case, please output -1.

Input Format

- line 1: n q
- line 2: b_1 b_2 $\ldots$ b_n
- line $2+i$ ($1 \leq i \leq q$): ℓ_i r_i

Output Format

- line 1: the minimum k such that after the first k events a might become b (or -1 if impossible).

Constraints

- $2 \leq n \leq 100\,000$.
- $1 \leq q \leq 1000$.
- $\{b_1, b_2, \ldots, b_n\}$ is a permutation of $\{1, 2, \ldots, n\}$.
- $b \neq [1, 2, \ldots, n]$.
- $1 \leq \ell_i < r_i \leq n$ for $i = 1, 2, \ldots, q$.
- All input values are integers.

Subtasks

1. (70 points) $n \leq 1000$.
2. (30 points) No additional constraints.

No.	Testdata Range	Time Limit (ms)	Memory Limit (KiB)
Samples	1-3	750	262144
1	1-23	750	262144
2	1-36	750	262144

Samples

Sample Input 1

```
5 3
3 4 2 5 1
1 4
4 5
2 4
```

This sample input satisfies the constraints of all the subtasks.

Sample Output 1

```
2
```

The shiny Noitatumrep a is initially $[1, 2, 3, 4, 5]$. After the first "エロ発生," $[a_1, a_2, a_3, a_4]$ gets shuffled. One of the possible results is $[3, 4, 2, 1, 5]$. If the second event happens, coincidentally, in such a way that the 4^{th} and the 5^{th} elements are swapped, the Noitatumrep will appear just like $b = [3, 4, 2, 5, 1]$.

Since a_5 will not be shuffled during the first event, it cannot result in b after just one "エロ発生." Thus the answer is 2.

Sample Input 2

```
5 3
5 4 3 2 1
1 4
4 5
2 4
```

This sample input satisfies the constraints of all the subtasks.

Sample Output 2

```
-1
```

We can observe that a_5 cannot be changed during the first event. From the second event onward, the shuffled positions do not contain 1, so 5 cannot be moved into position 1.

Sample Input 3

```
5 3
5 4 3 2 1
3 5
1 3
3 5
```

This sample input satisfies the constraints of all the subtasks.

Sample Output 3

```
-1
```

Problem E. Balanced Array

- 2023.10.06 15:00 Update: Strengthened testcases and rejudged solutions.

Problem Description

Given an array $a = [a[0], a[1], \ldots, a[n-1]]$ with $\sum_{i=0}^{n-1} a[i] = 0$, you want to balance the array such that every element becomes 0.

To achieve this, you can perform the following operations any number of times:

- Choose an index i $(0 \leq i \leq n-1)$, pay $w[i]$ dollars, and update $(a[i], a[(i+1) \bmod n])$ to $(a[i]-1, a[(i+1) \bmod n]+1)$.
- Choose an index i $(0 \leq i \leq n-1)$, pay $w[i]$ dollars, and update $(a[i], a[(i+1) \bmod n])$ to $(a[i]+1, a[(i+1) \bmod n]-1)$.

It is guaranteed that $w[n-1] = 0$. Can you determine the minimum amount of dollars you have to pay to balance the array?

Input Format

- line 1: n
- line 2: $a[0]$ $a[1]$ $\ldots$ $a[n-1]$
- line 3: $w[0]$ $w[1]$ $\ldots$ $w[n-1]$

Output Format

- line 1: the minimum cost to make the array balanced.

Constraints

- $2 \leq n \leq 1\,000\,000$.
- $|a[i]| \leq 1000$ for $i = 0, 1, \ldots, n-1$.
- $\sum_{i=0}^{n-1} a[i] = 0$.
- $0 \leq w[i] \leq 1000$ for $i = 0, 1, \ldots, n-1$.
- $w[n-1] = 0$.
- All the inputs are integers.

Subtasks

1. (25 points) $n \leq 100$; $|a[i]| \leq 100$ for $i = 0, 1, \ldots, n-1$.
2. (75 points) No additional constraints.

No.	Testdata Range	Time Limit (ms)	Memory Limit (KiB)
Samples	1-4	1000	262144
1	5-21	1000	262144
2	1-33	1000	262144

Samples

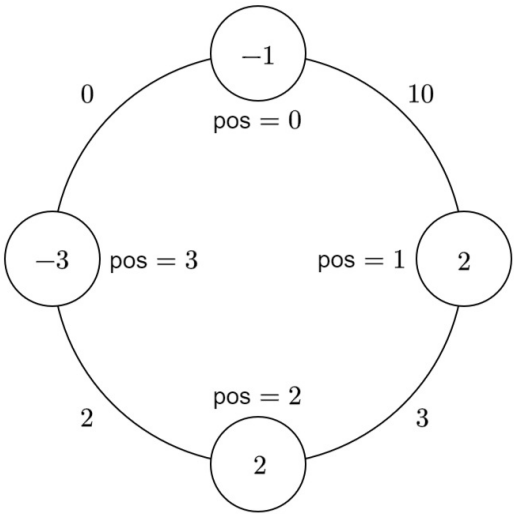
Sample Input 1

```
4
-1 2 2 -3
10 3 2 0
```

This sample input satisfies the constraints of all the subtasks.

Sample Output 1

```
14
```



The picture of sample 1.

The number inside the circle is $a[i]$ and the number on the line connecting position i and $(i+1) \bmod 4$ is $w[i]$.

In the optimal solution, we first pay $w[1] = 3$ dollars twice to update the array to $[-1, 0, 4, -3]$. Then, we pay $w[2] = 2$ dollars four times to further update the array to $[-1, 0, 0, 1]$. Finally, we pay $w[3] = 0$ dollars to achieve a balanced array.

Sample Input 2

```
5
4 1 -9 3 1
7 3 5 3 0
```

This sample input satisfies the constraints of all the subtasks.

Sample Output 2

```
58
```

Sample Input 3

```
8
10 20 30 40 -40 -30 -20 -10
88 46 90 17 22 63 99 0
```

This sample input satisfies the constraints of all the subtasks.

Sample Output 3

```
8290
```

Sample Input 4

```
4
1000 -1000 1000 -1000
0 0 0 0
```

This sample input satisfies the constraints of Subtasks 2.

Sample Output 4

```
0
```

Problem F. Path of the Mind

- 2023.10.06 15:00 Update: Fixed typo in Sample Output 4.

Problem Description

You opened Chapter 75 of "*Neko-chan Chronicle*," where the Brave Neko-chan Squad reached the final trial, the "*Path of the Mind*."

In the Path of the Mind, each squad member must confront their inner demons, strengthen their resolve, and overcome them to successfully reach the other side.

The Path of the Mind consists of k stages. While these stages may appear identical to observers, those within will sense subtle differences. Through the Brave Neko-chan Squad's pre-trial investigation, they found that each stage can be quantified by three values: *damage* $sd[i]$, *endurance* $se[i]$, and *property* $sp[i]$ $(0 \leq i \leq k-1)$. The "property" is the reason for the diverse nature of the Path of the Mind.

Simultaneously, the n brave warriors possess distinct attributes: *damage* $wd[j]$, *endurance* $we[j]$, and *property* $wp[j]$ $(0 \leq j \leq n-1)$. Battle with inner demons is akin to a debate: starting with a brave warrior, both sides take turns speaking, inflicting damage and depleting the opponent's endurance until one admits defeat (endurance ≤ 0).

The Brave Neko-chan Squad has its own interpretation of attributes: they believe attributes are the result of different personalities combined! Each personality has a "*prime*" influence, and moving towards the extreme of that personality increases the "*power*" of that prime. In other words, decomposing the quantified attributes into "*prime factorization*" reveals a person's character! When a brave warrior's attributes align with a stage's property, it indicates a profound impact, making the challenge more difficult. When a warrior with property p_w faces a stage with property p_s, they create an affinity of $f = \gcd(p_w, p_s)$, and both damage and endurance of that stage increase by f.

Since the trial grounds won't present insurmountable challenges, assistance is offered for the enigmatic Path of the Mind. Each member of the Brave Neko-chan Squad receives a shield with an energy level of a. When damaged, the shield automatically consumes energy to resist some of the damage (see below). This is both help and a test, examining the warriors' self-awareness of their abilities. Warriors can discuss and choose the level of assistance a they wish for their squad. While selecting $a = 10^{10^{100}}$ would ensure an easy passage, choosing non-optimal a could also lead to success without any rewards.

The shield will **automatically resist** attack damage as long as it has energy. If the opponent's damage is d_s and the current shield energy is a', the damage received becomes $\max(d_s - a', 0)$, and the shield's energy becomes $\max(a' - 1, 0)$.

As the leader of the Brave Neko-chan Squad, Neko-chan shoulders the mission of choosing the minimum assistance level a. Although her legendary story continues, the exact answer is not written, leaving you curious about how much help the Brave Neko-chan Squad ultimately received. You investigated the information about each warrior and each stage at that time. Now, find out the minimum value of a!

Oh, by the way, because Neko-chan's charisma is extraordinary, the squad consisted of tens of thousands of members at that time!

Input Format

- line 1: n k
- line 2: $wd[0]$ $wd[1]$ $\ldots$ $wd[n-1]$
- line 3: $we[0]$ $we[1]$ $\ldots$ $we[n-1]$
- line 4: $wp[0]$ $wp[1]$ $\ldots$ $wp[n-1]$
- line 5: $sd[0]$ $sd[1]$ $\ldots$ $sd[k-1]$
- line 6: $se[0]$ $se[1]$ $\ldots$ $se[k-1]$
- line 7: $sp[0]$ $sp[1]$ $\ldots$ $sp[k-1]$

Output Format

- line 1: the minimum possible value of a.

Constraints

- $1 \leq n \leq 100\,000$.
- $1 \leq k \leq 500$.
- $1 \leq wd[i], we[i] \leq 10^6$ for $i = 0, 1, \ldots, n-1$.
- $1 \leq wp[i] \leq 1000$ for $i = 0, 1, \ldots, n-1$.
- $1 \leq sd[i], se[i] \leq 10^6$ for $i = 0, 1, \ldots, k-1$.
- $1 \leq sp[i] \leq 1000$ for $i = 0, 1, \ldots, k-1$.
- All input values are integers.

Subtasks

1. (10 points) $k = 1$.
2. (10 points) $n = 1$.
3. (75 points) $n \leq 2000$.
4. (5 points) No additional constraints.

No.	Testdata Range	Time Limit (ms)	Memory Limit (KiB)
Samples	1-4	2500	262144
1	5-17	2500	262144
2	18-30	2500	262144
3	18-42	2500	262144
4	1-58	2500	262144

Samples

Sample Input 1

```
3 4
7 4 1000000
3 29 1
6 8 1
9 1 3 1000000
5 9 5 1
8 12 7 997
```

This sample input satisfies the constraints of Subtasks 3, 4.

Sample Output 1

```
10
```

Let us simulate how the warrior 1 with $(d_w, e_w, p_w) = (4, 29, 8)$ fight through their inner demons with assistance level 10:

(d_w, e_w, p_w)	Shield	Stage	(d_s, e_s, p_s)	Notes
$(4, 29, 8)$	10	0	$(9, 5, 8)$	The warrior enters stage 0.
$(4, 29, 8)$	10	0	$(17, 13, 8)$	Stage 0 gains $8 = \gcd(8, 8)$ damage and endurance.
$(4, 29, 8)$	10	0	$(17, 9, 8)$	Warrior attacks for 4 damage.
$(4, 22, 8)$	9	0	$(17, 9, 8)$	Stage 0 attacks for $7 = \max(17 - 10, 0)$ damage.
$(4, 22, 8)$	9	0	$(17, 5, 8)$	Warrior attacks for 4 damage.
$(4, 14, 8)$	8	0	$(17, 5, 8)$	Stage 0 attacks for $8 = \max(17 - 9, 0)$ damage.
$(4, 14, 8)$	8	0	$(17, 1, 8)$	Warrior attacks for 4 damage.

$(4, 5, 8)$	7	0	$(17, 1, 8)$	Stage 0 attacks for $9 = \max(17 - 8, 0)$ damage.
$(4, 5, 8)$	7	0	$(17, -3, 8)$	Warrior attacks for 4 damage. **Warrior wins!**
$(4, 5, 8)$	7	1	$(1, 9, 12)$	The warrior enters stage 1.
$(4, 5, 8)$	7	1	$(5, 13, 12)$	Stage 1 gains $4 = \gcd(8, 12)$ damage and endurance.
$(4, 5, 8)$	7	1	$(5, 9, 12)$	Warrior attacks for 4 damage.
$(4, 5, 8)$	6	1	$(5, 9, 12)$	Stage 1 attacks for $0 = \max(5 - 7, 0)$ damage.
$(4, 5, 8)$	6	1	$(5, 5, 12)$	Warrior attacks for 4 damage.
$(4, 5, 8)$	5	1	$(5, 5, 12)$	Stage 1 attacks for $0 = \max(5 - 6, 0)$ damage.
$(4, 5, 8)$	5	1	$(5, 1, 12)$	Warrior attacks for 4 damage.
$(4, 5, 8)$	4	1	$(5, 1, 12)$	Stage 1 attacks for $0 = \max(5 - 5, 0)$ damage.
$(4, 5, 8)$	4	1	$(5, -3, 12)$	Warrior attacks for 4 damage. **Warrior wins!**
$(4, 5, 8)$	4	2	$(3, 5, 7)$	The warrior enters stage 2.
$(4, 5, 8)$	4	2	$(4, 6, 7)$	Stage 2 gains $1 = \gcd(8, 7)$ damage and endurance.
$(4, 5, 8)$	4	2	$(4, 2, 7)$	Warrior attacks for 4 damage.
$(4, 5, 8)$	3	2	$(4, 2, 7)$	Stage 2 attacks for $0 = \max(4 - 4, 0)$ damage.
$(4, 5, 8)$	3	2	$(4, -2, 7)$	Warrior attacks for 4 damage. **Warrior wins!**
$(4, 5, 8)$	3	3	$(1\,000\,000, 1, 997)$	The warrior enters stage 3.
$(4, 5, 8)$	3	3	$(1\,000\,001, 2, 997)$	Stage 3 gains $1 = \gcd(8, 997)$ damage and endurance.
$(4, 5, 8)$	3	3	$(1\,000\,001, -2, 997)$	Warrior attacks for 4 damage. **Warrior wins!**

It can be shown that with an assistance level of 9, warrior 1 will lose to the "Path of the Mind." Neko-chan had to choose an assistance level of at least 10 to ensure the survival of the Brave Neko-chan Squad.

Sample Input 2

```
1 4
5
17
1
3 8 12 6
8 7 10 14
1 1 1 1
```

This sample input satisfies the constraints of Subtasks 2, 3, 4.

Sample Output 2

```
10
```

Sample Input 3

```
4 1
3 8 1 7
20 3 7 16
1 1 1 1
6
8
1
```

This sample input satisfies the constraints of Subtasks 1, 3, 4.

Sample Output 3

```
11
```

Sample Input 4

```
1 1
1
1
1
100000
100000
1000
```

This sample input satisfies the constraints of all the subtasks.

Sample Output 4

```
200000
```

The warrior cannot take any damage. Since the battle lasts for $100\,000$ rounds and the damage for stage 0 is $100\,001$, Neko-chan had to choose an assistance level of at least $200\,000$ for the warrior to survive.