Graduation Party (`confetti`)

After graduation, Marco gave a party to celebrate in style the end of his studies! As in every respectable party, he brought a lot of confetti of different types for his guests.

Figure 1: Some confetti with different colors and flavors.

The party is now over and Marco realized that he still has plenty of leftover confetti! He has already eaten lots of them and does not want to exaggerate, thus he had come up with a nice idea. He wants to distribute confetti to his friends but being careful not to make anyone unhappy.

In fact, he has C_0 confetti of one type, C_1 confetti of another type and so on... for a total of N different types. Marco wants to invite some friends and **give the same quantity of confetti to all of them** and this must hold **for all the types of confetti**. He also does not want to keep any for himself: if there are four confetti of a given type, he can invite only one, two or four friends to make a fair division. Help him: how many friends can he invite?

> ☞ Among the attachments of this task you may find a template file `confetti.*` with a sample incomplete implementation.

Input

The first line contains the only integer N. The second line contains N integers C_i, indicating how many confetti of type i are left.

Output

You need to write a single line with space-separated integers: **all** possible number of friends Marco can invite (listed in increasing order).

Constraints

- $1 \leq N \leq 100$.
- $1 \leq C_i \leq 10^{18}$ for each $i = 0 \ldots N-1$.
- You must output *all* the possible number of friends that can be invited *in increasing order*.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points)	Examples.
– **Subtask 2** (10 points)	$N = 1$, $C_i \leq 10^6$ for each $i = 0 \ldots N-1$.
– **Subtask 3** (30 points)	$C_i \leq 10^9$ for each $i = 0 \ldots N-1$.
– **Subtask 4** (40 points)	The highest number of people that can be invited is guaranteed to be at most 10.
– **Subtask 5** (20 points)	No additional limitations.

Examples

input	output
`3` `10 6 8`	`1 2`
`4` `24 30 6 125`	`1`

Explanation

In the **first sample case**, Marco can invite 1 friend (and give all confetti to her) or 2 friends (giving each five confetti of the first type, three of the second type and four of the last type). No other number works (for instance, with 3 friends Marco would not be able to fairly split the 10 confetti of the first type).

In the **second sample case**, the only feasible solution is to invite just one friend: every other number works for some types of confetti but not for others.

Forum Management (`forum`)

Edoardo is having an hard time maintaining the forum. In particular, the component responsible of sending to the users the notifications about new comments is very slow, so he would like to make it better. The forum can be described as a tree of directories, where each directory may contain other directories and zero or more posts. There are D directories, identified by integers from 0 to $D-1$, and P posts in total, identified by integers from D to $D+P-1$. The directory with identifier zero is the root of the tree; this means that every other directory and post is (directly or indirectly) contained in that directory. Edoardo knows that every directory or post with identifier i is directly contained inside the directory with identifier A_i. Since the root directory is not contained in any other directory, we define A_0 to be equal to -1.

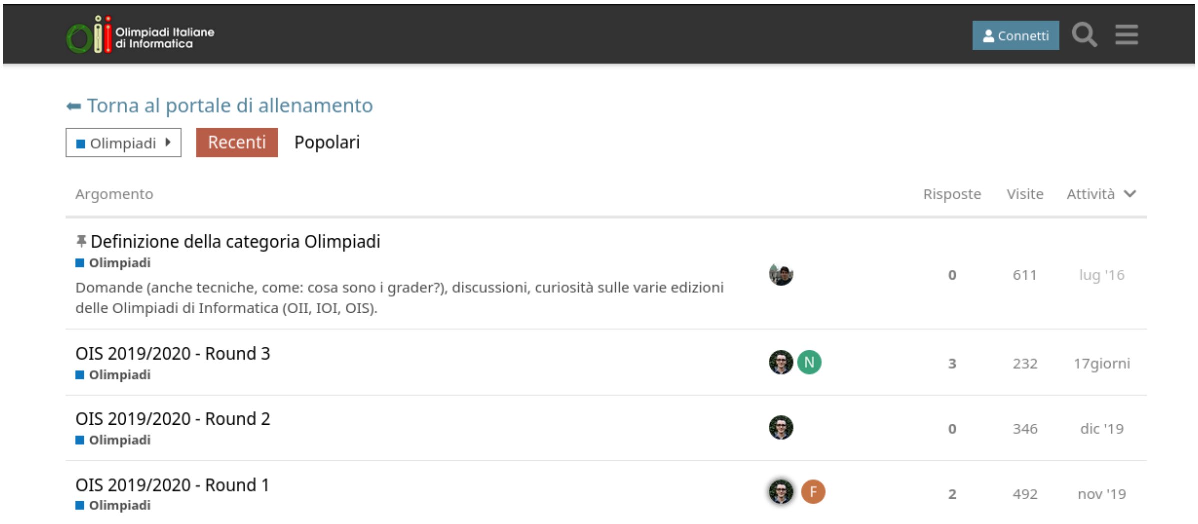


Figure 1: Our forum!

To make things easier, Edoardo considers only the E events triggered by one user. The j-th event has a type T_j and an identifier ID_j. The j-th event can be one out of three types:

- If $T_j = 0$, it means that the user unsubscribed from the directory or post with identifier ID_j. When a user unsubscribes from a directory, he unsubscribes recursively from all the directories and posts contained in it.
- If $T_j = 1$, then the user subscribed to the directory or post with identifier ID_j. Just like the previous case, when a user subscribes to a directory, he subscribes recursively to all the directories and posts contained in it.
- If $T_j = 2$, then a new comment has been published on post with identifier ID_j.

Edoardo has the list of all the events in chronological order, and he also knows that initially the user is not subscribed to any directory or post. Help Edoardo find out, for every event with $T_j = 2$, if the user is subscribed or not to the post with identifier ID_j, in order to decide whether to send the user a notification or not.

☞ Among the attachments of this task you may find a template file `forum.*` with a sample incomplete implementation.

Input

The first line contains three integers D, P and E, the number of directories, the number of posts and the number of events. The second line contains $D + P$ integers A_i, the i-th beign the parent directory of directory or post with identifier i. The following E lines contain the description of the various events, in chronological order. The j-th event is described by two integers T_j and ID_j, the type and identifier associated to the j-th event.

Output

You need to write a line with an integer for each event with $T_j = 2$: you have to write one if the user is subscribed to the post with identifier ID_j, zero otherwise.

Constraints

- $1 \leq D, P, E \leq 500\,000$.
- $A_0 = -1$.
- $0 \leq A_i < D$, for each $i = 1 \ldots D + P - 1$.
- $0 \leq T_j \leq 2$, for each $j = 0 \ldots E - 1$.
- $0 \leq ID_j < D + P$ if $T_j \neq 2$, otherwise $D \leq ID_j < D + P$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points) Examples.

– **Subtask 2** (10 points) $D, E \leq 100\,000$, $P = 1$ and $A[i] = i - 1$ for each $i = 0 \ldots D + P - 1$

– **Subtask 3** (10 points) $P, E \leq 100\,000$, $D = 1$

– **Subtask 4** (15 points) $D, E \leq 100\,000$, $P = 1$

– **Subtask 5** (20 points) $D, P, E \leq 1000$

– **Subtask 6** (15 points) $D, P, E \leq 100\,000$ and there are at most 10 events with $T_j = 2$

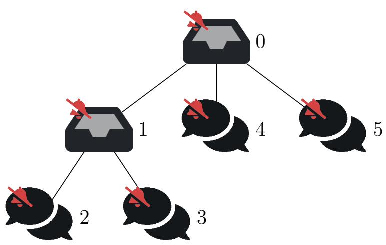
– **Subtask 7** (30 points) No additional limitations.

Examples

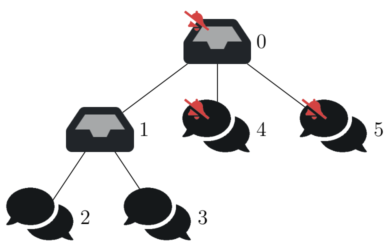
input	output
2 4 6 -1 0 1 1 0 0 1 1 2 3 2 4 0 0 2 2 2 5	1 0 0 0
3 2 7 -1 0 0 1 2 1 1 2 3 2 4 0 0 1 4 2 3 2 4	1 0 0 1

Explanation

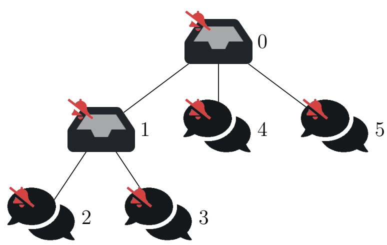
In the **first sample case** the first update subscribes to the directory 1 and the posts 2 and 3. The second update unsubscribes from everything.



Initial situation.

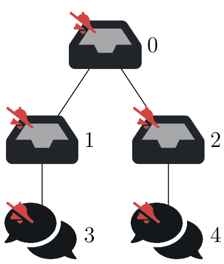


After the first update.

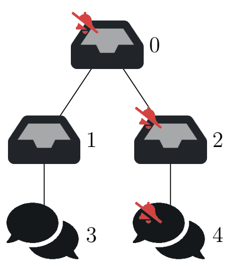


After the second update.

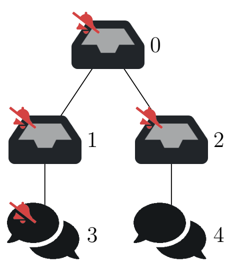
In the **second sample case** the first update subscribes to directory 1 and post 3, the second update unsubscribes from everything and the third subscribes to post 4.



Initial situation.



After the first update.



After the second and third update.

Pay That Box! (gameshow)

William got selected for the famous game show *Pay That Box!* He will start with a budget of M euros, then open a sequence of N boxes one at a time, from $i = 0$ to $N - 1$. Each box contains a prize, which he can *get* using P_i euros from his budget, or he can *pass* getting a compensation of C_i euros.

Figure 1: Boxes. . . what will be in there?

Since William is notoriously impulsive, he will pay for every box that he can afford, fearing that it may be his last chance to get a prize. Given the list of boxes that he will open, how many prizes will he be able to get, and how many euros will he have left by the end of the game?

> ☞ Among the attachments of this task you may find a template file `gameshow.*` with a sample incomplete implementation.

Input

The first line contains integers N, M. The second line contains N integers P_i. The third line contains N integers C_i.

Output

You need to write a single line two integers: the number of prizes that William will get, and how may euros will he spare at the end.

Constraints

- $1 \leq N \leq 100\,000$.
- $0 \leq M \leq 10^9$.
- $0 \leq P_i, C_i \leq 10^9$ for each $i = 0 \ldots N - 1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.
- **Subtask 2** (20 points) $M = 10^9$, $P_i \leq 10\,000$ for each i.
- **Subtask 3** (20 points) $M, C_i \leq 10\,000$, $P_i = 10^9$ for each i.
- **Subtask 4** (25 points) $N \leq 100$.
- **Subtask 5** (35 points) No additional limitations.

Examples

input	output

`4 10`
`42 23 35 17`
`13 55 11 7`
	`1 18`

`6 0`
`6 50 40 20 60 80`
`0 90 70 30 50 0`
	`3 0`

Explanation

In the **first sample case**, William cannot afford the first prize, thus he has to pass gaining 13 euros. He then starts the second round with $10 + 13 = 23$ euros, which are enough for the second prize, so he pays all of his money for it remaining without a budget. Afterwards, he has to pass on the third prize, gaining 11 euros. Those are not enough to pay for the fourth prize, so he will pass that also, ending the game with $11 + 7 = 18$ euros.

Notice that William could have followed better strategies, allowing him to get two prizes... but he can't help to be greedy!

In the **second sample case**, William will pay for the 3rd, 4th and 6th prizes.

Find the Treasure (islands)

Giorgio and Luca, while looking through old stuff in the basement of the OIS mansion, found a treasure map! However, the map has faded and is now impossible to understand the exact location of the hidden treasure. The map M can be represented as a two-dimensional grid with R rows and C columns, where each cell can either be land or sea. In particular, for the cell on the i-th row and j-th column, $M[i][j] = 1$ if the cell represents a portion of land, otherwise $M[i][j] = 0$ if it is part of the sea.

Figure 1: Edoardo's treasure map partially recovered with advanced techniques.

They want to find the treasure at all costs and are ready to start a trip to visit all *islands* on the map. An island I can be defined as a set of *land* cells satisfying these three conditions:

- **Maximality.** There are no land cells outside I sharing a side with a cell of I;
- **Connection.** It is possible to move from any cell of I to any other cell of I by repeatedly moving between side-adjacent cells of I;
- **Insularity.** No cell of I is on the border of the map (first or last row or column): those are just *peninsulas*, there's no point in wasting time on them.

Help Luca and Giorgio find the number of islands present in the map!

☞ Among the attachments of this task you may find a template file `islands.*` with a sample incomplete implementation.

Input

The first line contains two integers R and C, the number of rows and columns. Then R lines follow, each containing C values describing a row of the map. The j-th value of the i-th line represents $M[i][j]$, which is equal to 1 if the cell on the i-th row and j-th column is a portion of land, 0 otherwise.

Output

You need to write a single line with an integer: the number of islands present in the map.

Constraints

- $1 \le R, C \le 1\,000$.
- $M[i][j]$ is equal to zero or one, for each $i = 0 \ldots R-1, j = 0 \ldots C-1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points) Examples.

– **Subtask 2** (10 points) $R, C \le 3$.

– **Subtask 3** (15 points) $R = 3$.

– **Subtask 4** (15 points) $C = 3$.

– **Subtask 5** (20 points) There exist no two land cells which are side-adjacent.

– **Subtask 6** (25 points) No land cell is on the border of the map.

– **Subtask 7** (15 points) No additional limitations.

Examples

input	output
4 4 0 0 0 0 0 1 0 0 0 0 1 1 0 0 1 1	1
6 6 0 0 0 0 0 0 0 1 0 0 1 0 0 1 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0	3
7 7 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 1 0 0 0 1 0 0 1 0 1 0 1 0 0 1 0 0 0 1 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0	2

Explanation

In the **first sample case**, there is one island formed by the land cell in position $(1,1)$. The set of connected land cells in position $(2,2)$, $(2,3)$, $(3,2)$ and $(3,3)$ is not an island since at least one of those land cells is placed on the border.

In the **second sample case**, there are three islands. The first island is composed of the cells in position $(1,1)$, $(2,1)$, $(3,1)$ and $(2,2)$. The second island is composed of land cells $(1,4)$ and $(2,4)$. The third island is composed only of a single land cell in position $(4,3)$.

In the **third sample case**, there are two islands, one “inside” the other.

Building Mistakes (polygon)

As building a house is a hard process, you should seriously consider asking an engineer pieces of advice and plans. Despite this, Luca ignored Marco's warnings and decided to try building his house himself. Following the well-known proverb *"you cannot build the roof without the floor"*, he started from building the floor. He was so happy with the newly purchased tiles that he placed all the 100×100 tiles, forming a square, even before raising the walls!

Rare picture of Luca placing his tiles before realizing the mistake. 2020, colorized.

What a terrible mistake! Now he has to break and throw away some of his tiles in order to raise the planned walls, keeping only those completely inside the house. The original plan of the house has the shape of a polygon and luckily all the vertexes of this polygon coincide with a corner of a tile.

How many tiles will survive the raising of the walls?

☞ Among the attachments of this task you may find a template file `polygon.*` with a sample incomplete implementation.

Input

The first line contains the only integer F, the number of walls to build. The next F lines contain two integers each: X_i and Y_i, the coordinates of the vertexes of the polygon.

Output

You need to write a single line with an integer: the number of tiles that are completely inside the polygon.

Constraints

- $3 \leq F \leq 1000$.
- $0 \leq X_i, Y_i \leq 100$ for $0 \ldots F - 1$.
- The walls do not intersect, thus two walls share a point if and only if they are adjacent.
- The angle between two adjacent walls is not zero nor 360 degrees.
- All the coordinates are distinct.

Scoring

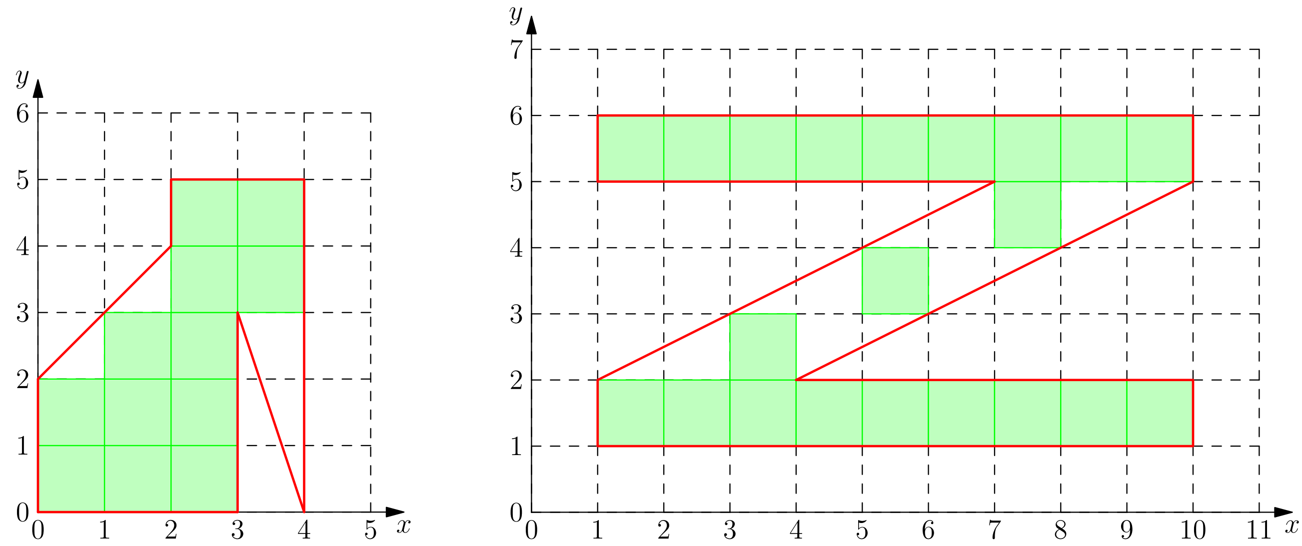
Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.
- **Subtask 2** (10 points) The polygon is a rectangle.
- **Subtask 3** (20 points) All the sides are vertical or horizontal.
- **Subtask 4** (20 points) $F = 3$ and $Y_0 = Y_1$.
- **Subtask 5** (20 points) The polygon is convex.
- **Subtask 6** (30 points) No additional limitations.

Examples

input	output
8 0 0 3 0 3 3 4 0 4 5 2 5 2 4 0 2	12
10 1 1 10 1 10 2 4 2 10 5 10 6 1 6 1 5 7 5 1 2	21

Explanation



Visualization of the two sample cases (first on the left, second on the right).

The images show only the relevant portion of the floor tiled with 100×100 tiles. Red lines represent the walls that need to be raised; the surviving tiles are highlighted in green.

A Tantrum with Consequences (raid)

President *Trample* just had one of his tantrums! Thus, he ordered an air raid of P airplanes against his favourite enemy, the *United Nations of Antarctica* (UNA).

Figure 1: Trample's airplanes ready to strike.

The UNA consists of a single line of N houses, each with a certain height H_i. The P airplanes will follow this line one at a time, dropping bombs from house $i = 0$ to $N-1$. Airplanes are not very precise, so they will only be able to hit the highest houses. More precisely, they only hit houses i whose height is strictly larger than that of both houses at its sides: $H_i > H_{i-1}$ and $H_i > H_{i+1}$.[1] When a house is hit, however, it will be burned to the ground, and H_i will become zero from that moment on, possibly enabling the **following planes** to hit more houses (not the current plane).

Giorgio, the newly elected emperor of the UNA, is rushing to calculate which houses will be hit, in order to help his fellow citizens to move to the safe houses.[2] How many houses will be burned to the ground after all P airplanes have passed?

☞ Among the attachments of this task you may find a template file `raid.*` with a sample incomplete implementation.

Input

The first line contains the two integers N and P. The second line contains N integers H_i.

Output

You need to write a single line with an integer: the number of houses that will be burned to the ground.

[1] The first and last house only need to be taller than their single neighbouring house.
[2] Waiting in open air it is not very comfortable in the UNA, as you may imagine.

Constraints

- $2 \leq N \leq 100\,000$.
- $1 \leq P, H_i \leq 10^6$ for each $i = 0 \ldots N-1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

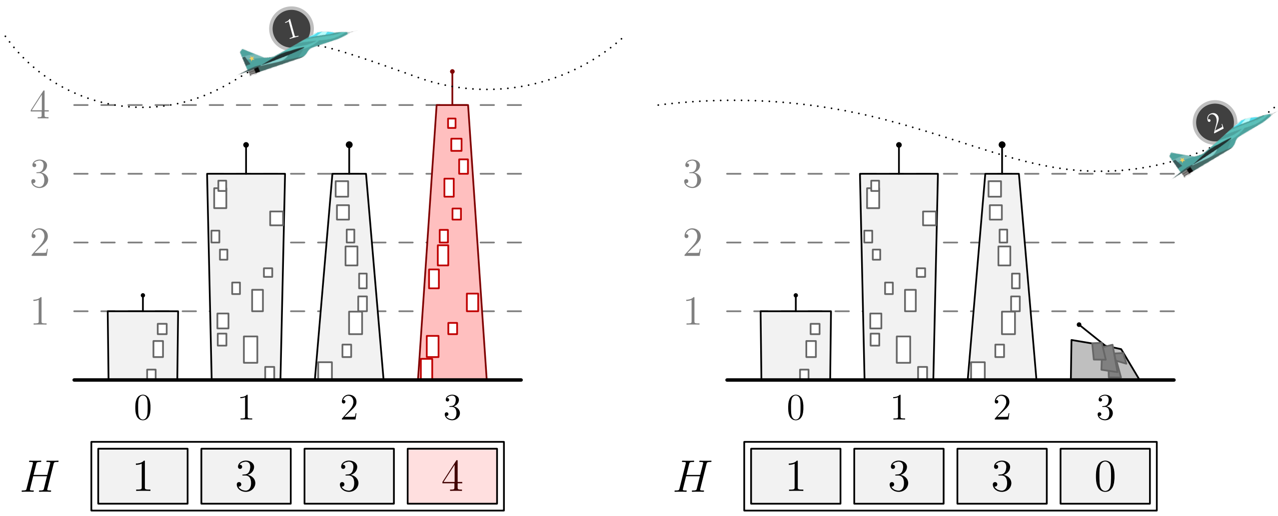
- **Subtask 1** (0 points) Examples.
- **Subtask 2** (10 points) $H_i = i + 1$.
- **Subtask 3** (15 points) $N, P \leq 10$.
- **Subtask 4** (20 points) $N, P \leq 1000$.
- **Subtask 5** (15 points) $N \leq 1000$.
- **Subtask 6** (40 points) No additional limitations.

Examples

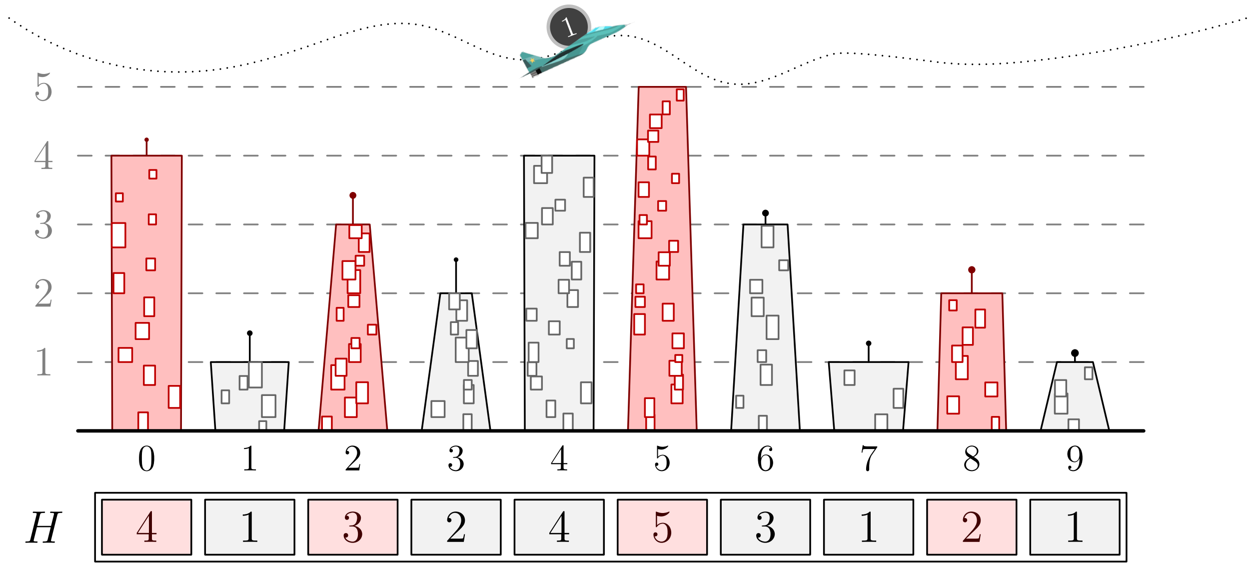
input	output
`4 10` `1 3 3 4`	`1`
`10 2` `4 1 3 2 4 5 3 1 2 1`	`8`

Explanation

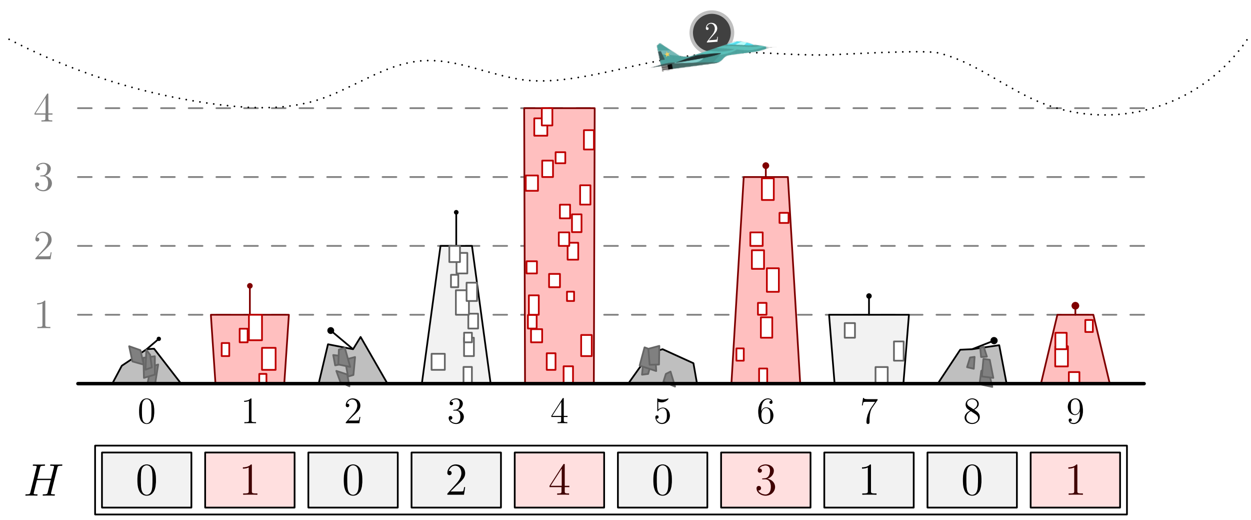
In the **first sample case**, the first plane destroys house 4, then the following planes cannot do anything.



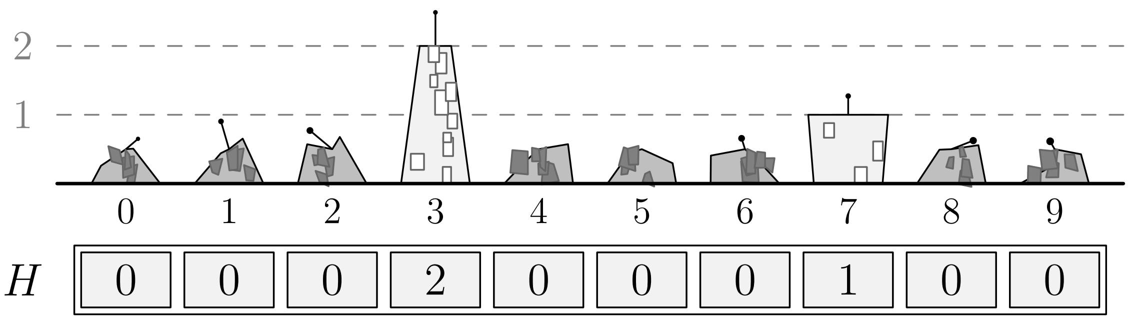
In the **second sample case**, the first plane can destroy 4 different houses.



Then, the second plane destroys 4 additional houses.



There are no other planes, so two houses will still stand.



Ropes Escape (`ropes`)

Edoardo was hacking a mainframe when he was traced by the cyberpolice (he didn't disconnect quickly enough). He is now in jail... but already planning his escape!

In fact, Edoardo managed to get a hold on *The Whip*, an evil tool that his captors regularly use to punish prisoners. The rope is made by a handle and a set of N interconnected ropes which form a tree: at the root of this tree we find the whip's handle (numbered 0), and each rope $i = 1 \dots N$ is directly tied to another rope P_i called *parent rope* (or to the handle if $P_i = 0$).

Figure 1: What *The Whip* looks like.

Edoardo's plan is to *untie* some of the ropes (one by one) from their parent rope and then *tie them back* to some other rope on the whip in order to form a single, very long rope (the handle should remain on one end of the rope). This rope can then be used to get out of the jail, as many prisoners did with bed sheets over the course of history!

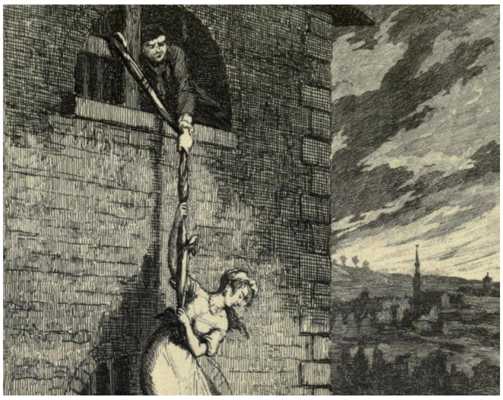


Figure 2: Prisoners escaping using bed sheets.

It's important to note that Edoardo can disconnect a rope from its parent even if there are other ropes connected to it or to the parent: he will then obtain a "sub-whip" that can then be reattached anywhere in the original whip.

Unfortunately, the ropes are tied together very strongly (after all those years of frantic usage) and untying rope i from its parent requires T_i seconds. Tying back a rope requires no time in comparison. Help Edoardo plan his escape by computing the minimum total time needed to transform *The Whip* into an escape rope!

> ☞ Among the attachments of this task you may find a template file `ropes.*` with a sample incomplete implementation.

Input

The first line contains the only integer N. The second line contains the N integers P_i. The third line contains the N integers T_i.

Output

You need to write a single line with an integer: the total time required for turning the whip into a straight rope that Edoardo can use to escape.

Constraints

- $1 \le N \le 100\,000$.
- $0 \le P_i \le N$ for each $i = 1 \ldots N$.
- $0 \le T_i \le 1000$ for each $i = 1 \ldots N$.
- All ropes are ultimately tied to the handle (*The Whip* is a single connected piece).

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points) Examples.

– **Subtask 2** (10 points) All ropes are directly tied to the whip's handle: $P_i = 0$ for all i.

– **Subtask 3** (20 points) $N = 3$.

– **Subtask 4** (30 points) Untying times are fixed: $T_i = 1$ for all i.

– **Subtask 5** (20 points) $N \le 10$.

– **Subtask 6** (20 points) No additional limitations.

Examples

input	output

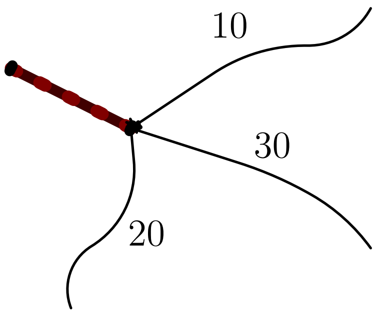
`3`
`0 0 0`
`10 30 20`
	`30`

`3`
`0 1 0`
`10 10 100`
	`10`

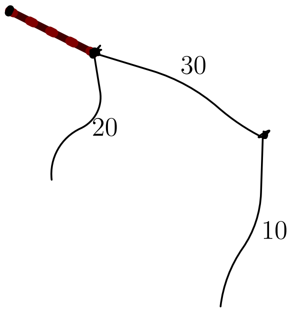
Explanation

In the **first sample case**, all ropes are directly connected to the handle. The best strategy is to:

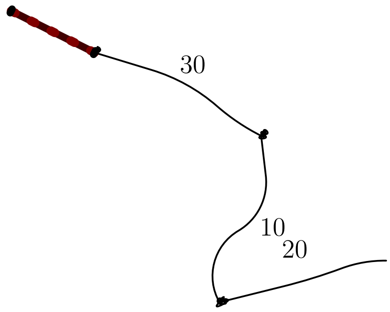
- disconnect rope 1 from the handle,
- connect rope 1 to rope 2,
- disconnect rope 3 from the handle,
- connect rope 3 to rope 2.



Initial situation.

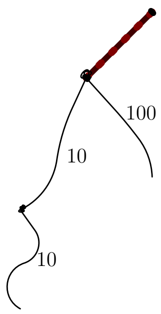


Untie rope 1 and connect to 2: 10 seconds.

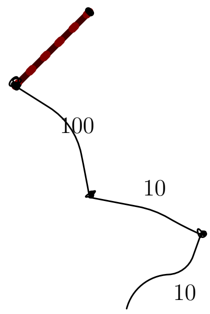


Untie rope 3 and connect to 1: 20 seconds.

In the **second sample case** we can make a straight rope by disconnecting rope 1 from the handle (obtaining the sub-whip of ropes 1 and 2), and then reconnect it to rope 3, in a total of 10 seconds.



Initial situation.



Untie rope 1 and connect to 2: 10 seconds.

Late for Work (`time`)

William has just woken up, but he forgot to set the alarm before going to bed, so he is now late for work! However, every cloud has a silver lining: he really needed a long sleep and now he is well rested.

William wants to know how much time he has slept. He looked at the digital clock he has on the nightstand before going to bed, so he knows the hour H_0 and minute M_0 he fell asleep (for example, if he fell asleep at `1:35`, H_0 is going to be equal to 1 and M_0 to 35). He also knows that he woke up at the hour H_1 and minute M_1 (for example, if he woke up at `10:15`, H_1 is equal to 10 and M_1 to 15). He is also sure that he has slept for less than 24 hours.

Given the time when William fell asleep and the time when he woke up, help him find how much time he has slept.

> ☞ Among the attachments of this task you may find a template file `time.*` with a sample incomplete implementation.

Input

The first line contains two integer H_0 and M_0, the hour and minute when William went to bed. The second line contains two integer H_1 and M_1, the hour and minute when William woke up.

Output

You need to write a single line with two integers: the time that William spent sleeping in hours and minutes. The number of minutes cannot be greater or equal to 60.

Constraints

- $0 \le H_0, H_1 < 24$.
- $0 \le M_0, M_1 < 60$.
- The time when William woke up is different from the time when he fell asleep.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points) Examples.

– **Subtask 2** (10 points) $H_0 = 0$ and $M_0 = 0$.

– **Subtask 3** (10 points) $H_0 = 0$ and $H_1 = 0$.

– **Subtask 4** (20 points) $H_0 < H_1$.

– **Subtask 5** (60 points) No additional limitations.

Examples

input	output
0 30 8 25	7 55
22 30 7 15	8 45

Explanation

In the **first sample case**, William fell asleep at half past midnight, and woke up at twenty-five past eight AM. This means that he has slept for 7 hours and 55 minutes.

In the **second sample case**, William fell asleep at half past ten PM, and he woke up at a quarter past seven AM. William has slept for 8 hours and 45 minutes.