





## Dangerous Flowers (antennas)

After yet another fight with his girlfriend, William knows very well that there is only one way to be forgiven: bring her a bouquet of fresh *edelweiss* flowers, her favourites. These flowers can be easily found in the mountains where William lives, however, you are not allowed to pick them unless you have a special permit. But there's no time for bureaucracy after such a fight, so he gotta take the risk and pick them without it!



Figure 1: *Leontopodium nivale*, commonly known as edelweiss.

Unfortunately, after picking up the very first flower, he triggered an unexpected alarm! By quickly hacking the alarm control unit, William realised that the warning is going to be transmitted to the forestry police by a line of  $N$  directional antennas in the mountains. Each antenna has a receiver with a threshold level of  $L_i$  decibel, and is able to transmit to following antennas with a power of  $P_i$  decibels. The signal decreases in strength by  $D$  decibel while travelling the distance between any two consecutive antennas. For example, if we have the following configuration of  $N = 4$  antennas with a decay  $D = 4$ :

				
L:	5	6	3	0
P:	10	7	5	8
$i$ :	0	1	2	3

In this case, the signal from antenna 0 is received by antenna 1 (where arrives with strength 6), but not from the subsequent ones (where arrives with strength 2 and  $-2$  respectively); the signal from antenna 1 is received by antenna 2; and the signal from antenna 2 is received by antenna 3. With the same configuration but a decay  $D = 3$ , the signal from antenna 1 would have been received by all the following ones, and similarly for the others.

Since the antennas need to run with a very low energy consumption, they are always receiving signals, but are transmitting only at regular intervals of  $T_i$  milliseconds starting from  $S_i$ : thus, the transmitting times are  $S_i$ ,  $S_i + T_i$ ,  $S_i + 2T_i$  and so on. If an antenna receives a signal at a certain time  $t$ , it is able to transmit it at its first transmitting times **strictly after**  $t$ .

Help William by computing when the warning is going to be received by the forestry police (which listens directly on the antenna  $N - 1$ ) so he can plan when to run away with the flowers he got!

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

## Input

The first line contains the two integers  $N, D$ . The following  $N$  lines contains four integers  $L_i, P_i, S_i, T_i$ .

## Output









You need to write a single line with an integer: the earliest time in which the last antenna will transmit the warning. If no such time exist (the warning will not be received) then you should write  $-1$ .

## Constraints

- $2 \leq N \leq 100\,000$ .
- $0 \leq D \leq 10\,000$ .
- $-10^9 \leq L_i < P_i \leq 10^9$  for each  $i = 0 \dots N - 1$ .
- $0 \leq S_i < T_i \leq 10\,000$  for each  $i = 0 \dots N - 1$ .

## 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.

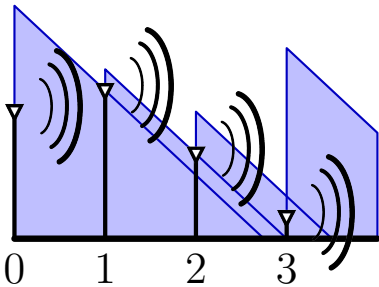
- |                                                                                     |                                                          |
|-------------------------------------------------------------------------------------|----------------------------------------------------------|
| – Subtask 1 (0 points)                                                              | Examples.                                                |
|  |                                                          |
| – Subtask 2 (10 points)                                                             | $N = 2$ .                                                |
|  |                                                          |
| – Subtask 3 (10 points)                                                             | $P_i = 10\,000, L_i = 0$ for all $i$ and $D = 10\,000$ . |
|  |                                                          |
| – Subtask 4 (10 points)                                                             | $P_i = 10\,000, L_i = 0$ for all $i$ .                   |
|  |                                                          |
| – Subtask 5 (20 points)                                                             | $P_i = 10\,000$ for all $i$ .                            |
|  |                                                          |
| – Subtask 6 (25 points)                                                             | $N \leq 500$ .                                           |
|  |                                                          |
| – Subtask 7 (15 points)                                                             | $D = 0$ .                                                |
|  |                                                          |
| – Subtask 8 (10 points)                                                             | No additional limitations.                               |
|  |                                                          |

Examples

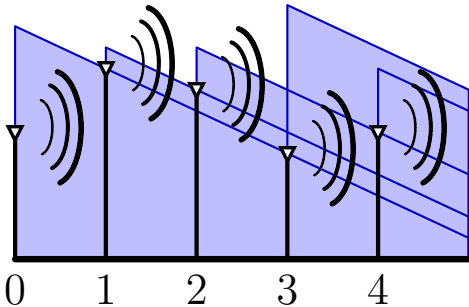
input	output
4 4 5 10 5 20 6 7 2 10 3 5 12 15 0 8 0 1	28
5 2 5 10 2 4 8 9 5 15 7 9 1 7 4 11 2 20 5 8 4 6	10
5 4 2 8 4 7 5 9 2 3 0 7 3 8 5 8 5 9 1 5 1 6	-1

Explanation

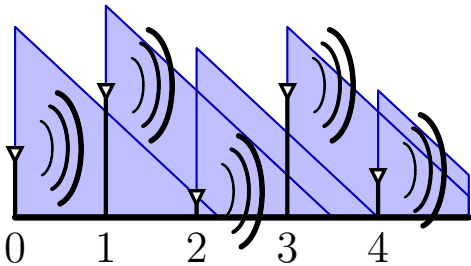
In the **first sample case**, antennas are configured as explained in the task statement (see also the picture below, where the height of the antenna represents its threshold, and the blue triangle represents the area where the power is enough). Every antenna transmits to the following one at time 5, 12, 27 and 28 respectively.



In the **second sample case**, antenna 0 is received by antennas 1 and 3; antenna 1 is received by antennas 2 and 3; antenna 2 is received by antennas 3 and 4; antenna 3 is received by antenna 4. Antennas transmit the warning for the first time respectively at 2, 5, 8, 22, 10.



In the **third sample case**, antennas transmit for the first time respectively at 4, never, 11, never, never. Notice that the fourth antenna could be able to communicate with the fifth, but it cannot receive from any of the previous, so the warning is not received and the answer is  $-1$ .



## Worst Exam Ever! (bonus)

After long weeks of waiting, Edoardo has finally got back the results of the last exam he took. The exam was made of  $N$  exercises, the  $i$ -th exercise had a maximum score of  $P_i$  points and he scored  $S_i$  points. Edoardo is not happy of the grade he received, and is considering to take it again. However, the professor doesn't want yet another exam to grade... and thus he decided to give him a generous *bonus*!

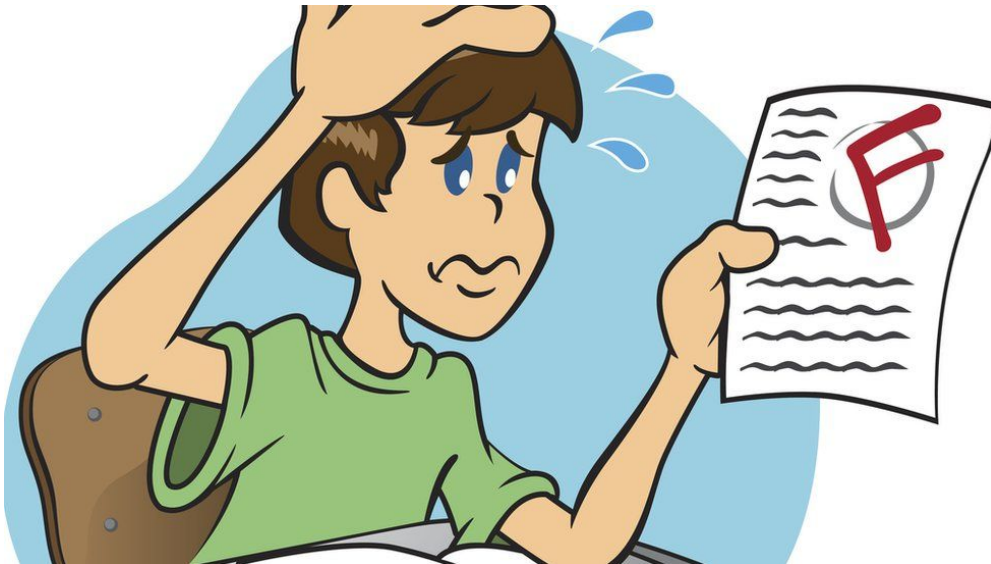


Figure 1: Edoardo is desperate!

More precisely, Edoardo can choose exactly one of the  $N$  exercises and that exercise will be discarded and will not take part in the grading of the exam. Note that the grade of the exam is defined as the ratio between the sum of points scored and the sum of maximum scores (excluding in both sums the exercise discarded, thanks to the generous bonus). Help Edoardo find which exercise to discard in order to get the best grade possible!

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

📎 Notice that you may find the wrong solution if you use floating-point variables to compare fractional values, due to precision errors. To find out if  $\frac{A}{B} \geq \frac{C}{D}$  you can check instead if  $A \cdot D \geq C \cdot B$ , while being careful that even if the input values fit inside a 32-bit integer, the intermediate multiplications may not. Then, use 64-bit integers such as `long long` for C and C++ and `Int64` for Pascal (the templates attached already do it).

## Input

The first line contains the only integer  $N$ . The following  $N$  lines contain two integers  $S_i, P_i$  each.

## Output








You need to write a single line with an integer: the index (starting from zero) of the exercise to be discarded. In case more than one solution exist, print one of them.

## Constraints

- $2 \leq N \leq 100\,000$ .
- $1 \leq P_i \leq 10\,000$  for each  $i = 0 \dots N - 1$ .
- $0 \leq S_i \leq P_i$  for each  $i = 0 \dots N - 1$ .

## 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** (15 points)       $N \leq 10, P_i \leq 10$  for every  $0 \leq i \leq N - 1$ .  

- **Subtask 3** (15 points)       $N \leq 1000, P_i \leq 10$  for every  $0 \leq i \leq N - 1$ .  

- **Subtask 4** (30 points)       $N \leq 5000$ .  

- **Subtask 5** (10 points)       $P_i = P_j$  for every  $0 \leq i, j \leq N - 1$   

- **Subtask 6** (10 points)       $S_i = S_j$  for every  $0 \leq i, j \leq N - 1$   

- **Subtask 7** (20 points)      No additional limitations.  


## Examples

input	output
3 1 1 2 2 1 3	2
3 1 1 1 2 1 2	1

## Explanation

In the **first sample case**, the best choice is to discard the last exercise, obtaining a grade of  $\frac{1+2}{1+2} = \frac{3}{3} = 1$ .

In the **second sample case**, Edoardo can choose to discard one of the last two exercises, obtaining a grade of  $\frac{1+1}{1+2} = \frac{2}{3}$ . This is the best grade that Edoardo can obtain.

## Orderly Evacuation (evacuation)

The *OIS control room*, where the staff prepares tasks for the contests, consists of  $N$  workstations (numbered from 0 to  $N - 1$ ) each assigned to a staff member (so that workstation  $i$  is assigned to staff member  $i$ ). Whenever the room alarm signals that lunch is ready, the whole staff rushes for the exit of the room, which is located by workstation 0, while orderly following the evacuation plan.




Figure 1: The OIS control room.

The evacuation plan of the room has a tree shape, so that anyone in workstation  $i$  has only one option for going towards the exit, which is workstation  $E_i$  (we say that  $E_0 = -1$  to represent that workstation 0 is connected to the exit). However, multiple workstations  $i, j$  may compete for the same  $E_i = E_j$ , forcing the staff members to use their *arrogance* to move forward. More precisely, every staff member  $i$  has an arrogance score  $A_i$ , and the evacuation proceeds as follows:

- The staff member in workstation 0 exits the room.
- The empty space in workstation 0 is immediately taken by the **most arrogant** staff member competing for it.
- The empty space left by that staff member is then taken by the most arrogant staff member competing for it, and so on until no empty space with competing members exists.
- The new staff member in workstation 0 exits the room, and the process starts over until everybody has left the room.

Giorgio, which is cooking today's lunch, would like to optimise the kitchen service by sorting the lunch boxes according to the staff members' preferences. Help him by predicting the order in which the staff members will exit the room!

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

### Input

The first line contains the only integer  $N$ . The second line contains  $N$  integers  $E_i$ . The third line contains  $N$  integers  $A_i$ .

## Output

You need to write a single line with  $N$  integers: the indices of staff members in exit order.

## Constraints

- $1 \leq N \leq 1\,000\,000$ .
- $E_0 = -1$  and  $0 \leq E_i \leq N - 1$  for each  $i = 1 \dots N - 1$ .
- Everybody is able to evacuate following the plan.
- $0 \leq A_i \leq 10^9$  for each  $i = 0 \dots N - 1$ .
- Arrogance scores are all 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)       $N \leq 10$ .
- Subtask 3 (15 points)      The plan is a line ( $E_i \neq E_j$  for each  $i, j = 0 \dots N - 1$ ).
- Subtask 4 (15 points)      The plan is a star ( $E_i = 0$  for each  $i = 1 \dots N - 1$ ).
- Subtask 5 (20 points)      There are at most 10 people in terminal positions.
- Subtask 6 (15 points)      The evacuation path for each position is no longer than 10.
- Subtask 7 (25 points)      No additional limitations.

## Examples

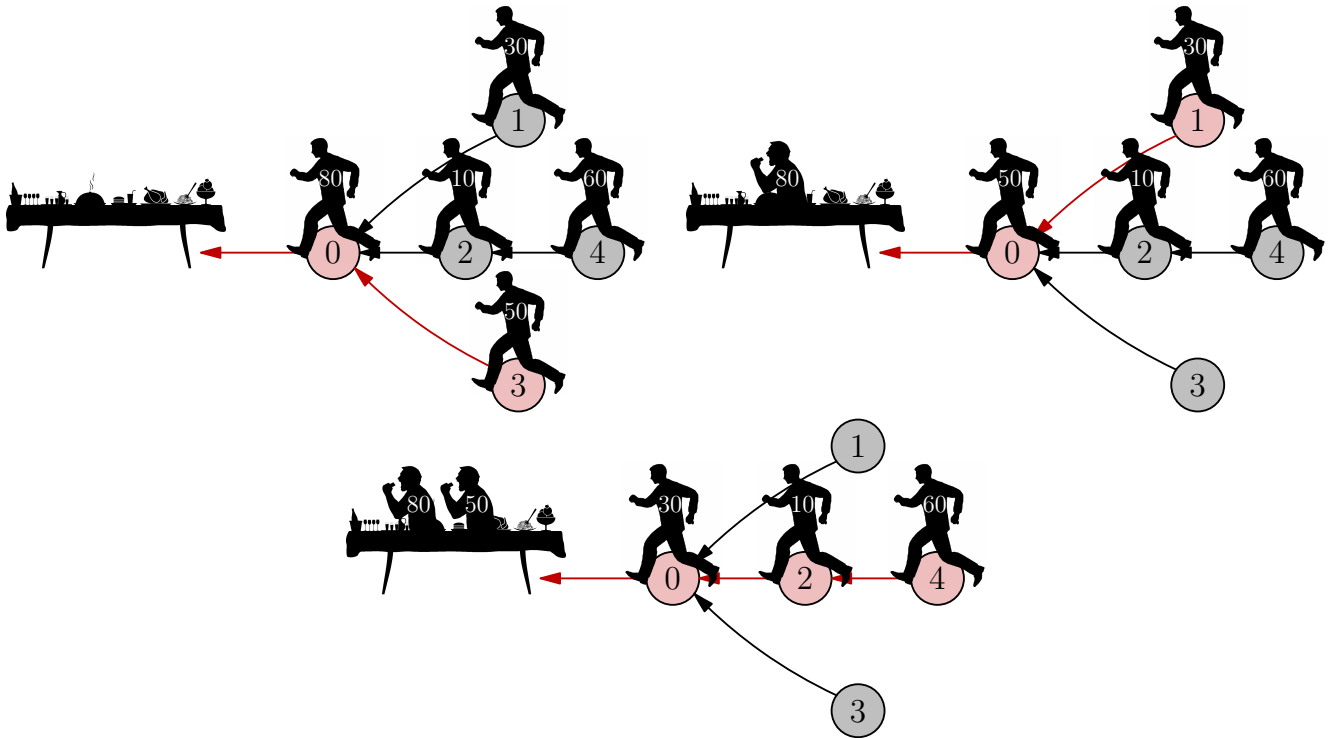
input	output
5 -1 0 0 0 2 80 30 10 50 60	0 3 1 2 4
7 -1 0 0 1 1 2 2 0 20 30 40 10 15 25	0 2 6 1 3 5 4



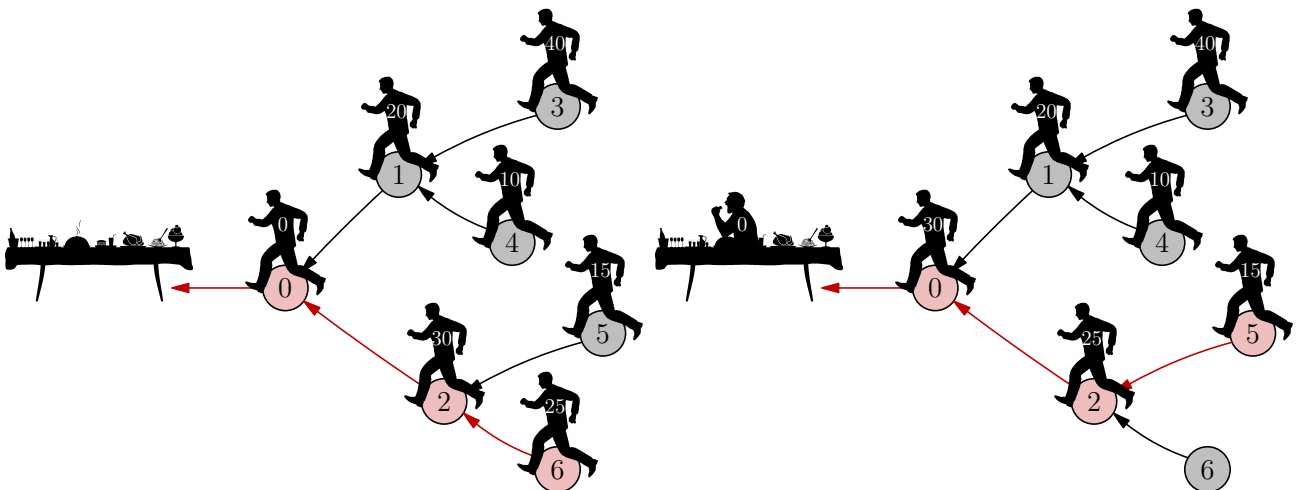
## Explanation

In the **first sample case**, the staff members proceed as follows.

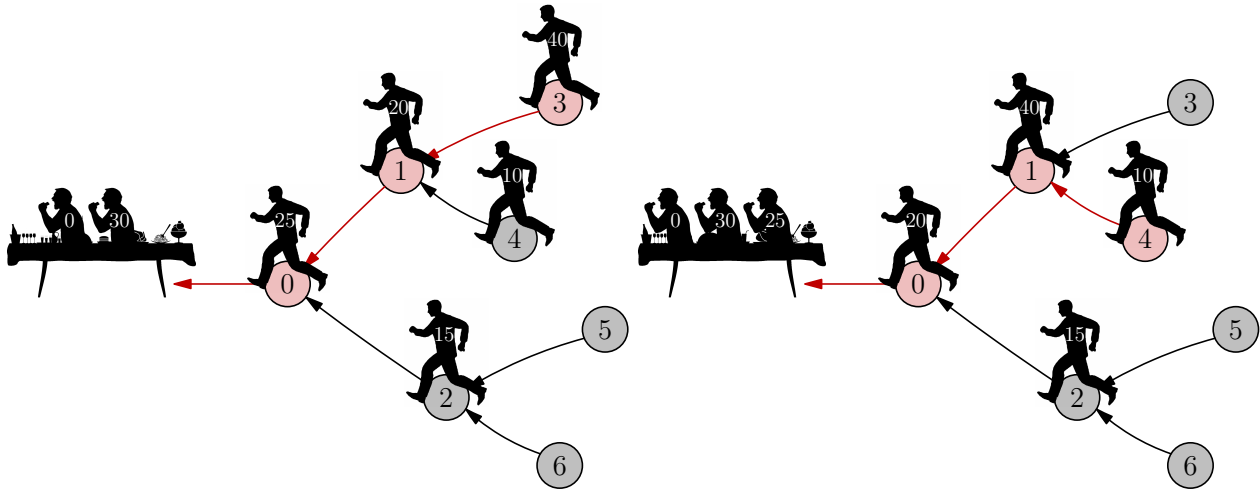
First, staff member 0 at workstation 0 immediately exits; leaving its place to staff member 3 at workstation 3 (which is the most arrogant contender). Then, staff member 3 at workstation 0 exits leaving its place to staff member 1 at workstation 1, which has now become the most arrogant contender for workstation 0. At this point, people are all in line and can exit in order.



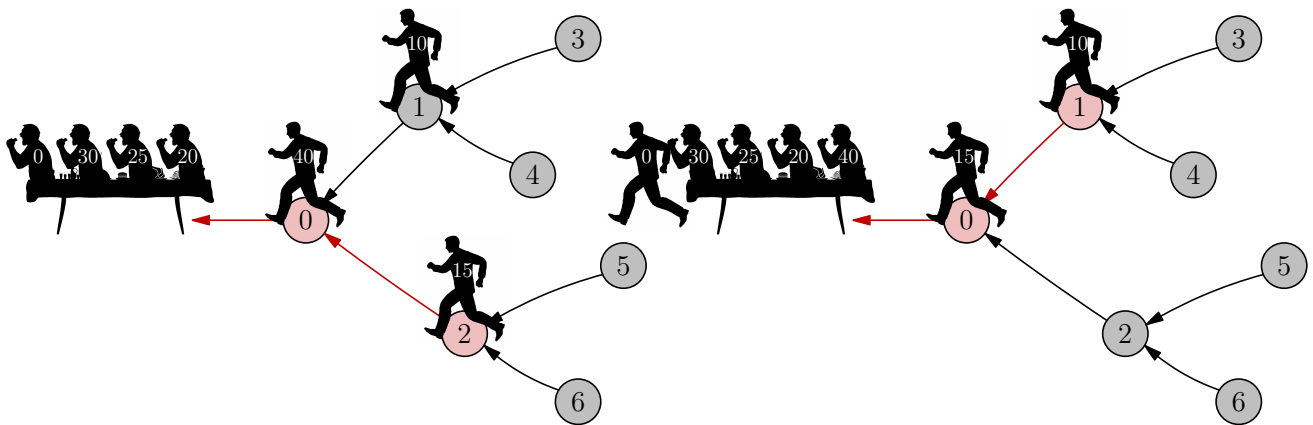
In the **second sample case**, the staff members proceed as follows.



Staff member 0 at workstation 0 immediately exits. For that workstation, two staff members of arrogance 20 and 30 are competing: the one with arrogance 30 takes precedence and can move there. His workstation is contended between staff members 5 and 6, with arrogance levels 15 and 25: the latter takes its place. Proceeding with the evacuation, staff member 2 at workstation 0 can now exit. Workstation 0 is now competed by two staff members: number 1 (arrogance 20) and number 6 (arrogance 25). Thus, number 6 can take the position at workstation 0 and will be the third to exit. His place will be taken over by member number 5, who is the only one left that competes for workstation 2.



The fourth to reach workstation 0 will then be member number 1 (arrogance 20) who wins over member number 5 (arrogance 15). His position will be taken over by staff member number 3 (arrogance 40). The fifth to exit will be him, as he wins the battle with number 5 (arrogance 15). At the end, we are left with members number 5 (arrogance 15) and 4 (arrogance 10) who will exit exactly in this order.



## Cultural Events (events)

Luca recently moved to Lugano, Switzerland, to complete his graduate studies. He is eager to discover the cultural scene of the city and he is constantly looking for cool events to participate in!



Figure 1: A night picture of the LAC (Lugano Arte & Cultura) building.

His main concern is about the prices of the tickets for those events, which are often not very cheap (like almost everything in Switzerland!). The university comes to the rescue, as it offers some vouchers to attend events for free.

You can buy the ticket for a certain event if you use a voucher whose value is high enough (at least equal to or greater than the price of the event). Once you use a voucher to buy a ticket, **it cannot be used anymore**.

Luca made a list of  $N$  interesting events with their prices and, so far, has collected  $V$  vouchers of different values from his university. He has no particular preference regarding the kind of event; he just wants to participate in as many as possible and asks you to help him.

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

### Input

The first line contains two integers  $N$  and  $V$ , respectively the number of events and the number of vouchers.

The second line contains  $N$  integers, where the  $i$ -th is the price (in CHF) of the ticket for the  $i$ -th event.

The third line contains  $V$  integers, where the  $i$ -th is the value (in CHF) of the  $i$ -th voucher.

### Output





You need to write a single line with an integer: the maximum number of events that Luca can attend.

## Constraints

- $1 \leq N, V \leq 50\,000$ .
- The price of events' tickets and the value of vouchers is between 1 and 1 000 000 (CHF) inclusive.

## 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** (15 points)      Every voucher has a value higher than **all** tickets' prices.  

- **Subtask 3** (30 points)       $N, V \leq 9$ .  

- **Subtask 4** (55 points)      No additional limitations.  


## Examples

input	output
3 2 25 10 12 15 25	2
4 4 100 80 20 35 40 200 10 10	2

## Explanation

In the **first sample case** Luca has only two vouchers. He can use the first one (15 CHF) for buying the ticket of the last event (12 CHF) and the second one (25 CHF) to buy the ticket of the first event.

In the **second sample case** Luca can only attend two events. A possible strategy is to use the 200 CHF voucher to buy the 80 CHF ticket and the 40 CHF voucher to buy the 35 CHF ticket. Other strategies are also possible, but in no case he can attend to three events.

## Exam Room (examroom)

Giorgio, famous university teacher, tomorrow is going to do the surveillance during his exam of Competitive Programming. In fact, even though the students are usually very honest, there is always someone that tries to cheat. To prevent it, Giorgio is studying a disposition for the students in the room that maximizes the distance between each student.



The exam room is composed by  $R \times C$  seats arranged in a rectangle of  $R$  rows and  $C$  columns. Giorgio wants his students to be arranged in rows and columns as well, but skipping some rows and some columns to space them out. Given a distance factor  $K$ , each student must be at least  $K$  seats away ( $K - 1$  seats in between) from the ones in his row and column.

Help him find the maximum number of students that fit inside the classroom.

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

The answer may be too large for a 32-bit integer: use 64-bit integers such as `long long` for C and C++ and `Int64` for Pascal (the templates attached already do it).

### Input

The first line contains three integers,  $R$ ,  $C$  and  $K$ .

### Output








You need to write a single line with an integer: the maximum number of students that fit in the room.

### Constraints

- $1 \leq R, C, K \leq 10^9$ .

### 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.

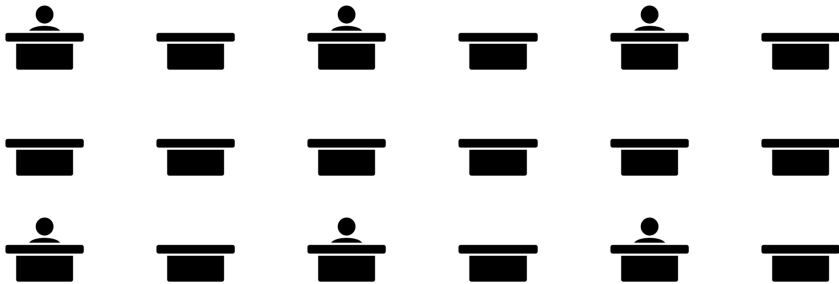
- **Subtask 1** (0 points)      Examples.  

- **Subtask 2** (5 points)       $K = 1$ .  

- **Subtask 3** (30 points)       $K \leq 2$ .  

- **Subtask 4** (10 points)       $R, C, K \leq 10$ .  

- **Subtask 5** (10 points)       $R, C, K \leq 100\,000$ .  

- **Subtask 6** (20 points)       $R$  and  $C$  are multiples of  $K$ .  

- **Subtask 7** (25 points)      No additional limitations.  


### Examples

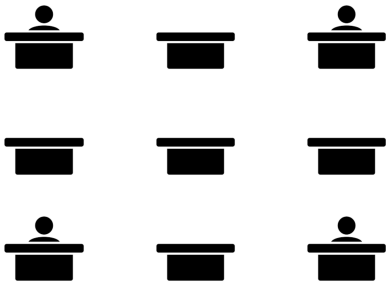
input	output
3 6 2	6
3 3 2	4

### Explanation

In the **first sample case** there are 3 rows and 6 columns, the students can fit in the first and last row but only on 3 columns, keeping at least one column free between them. This arrangement is shown in this image:



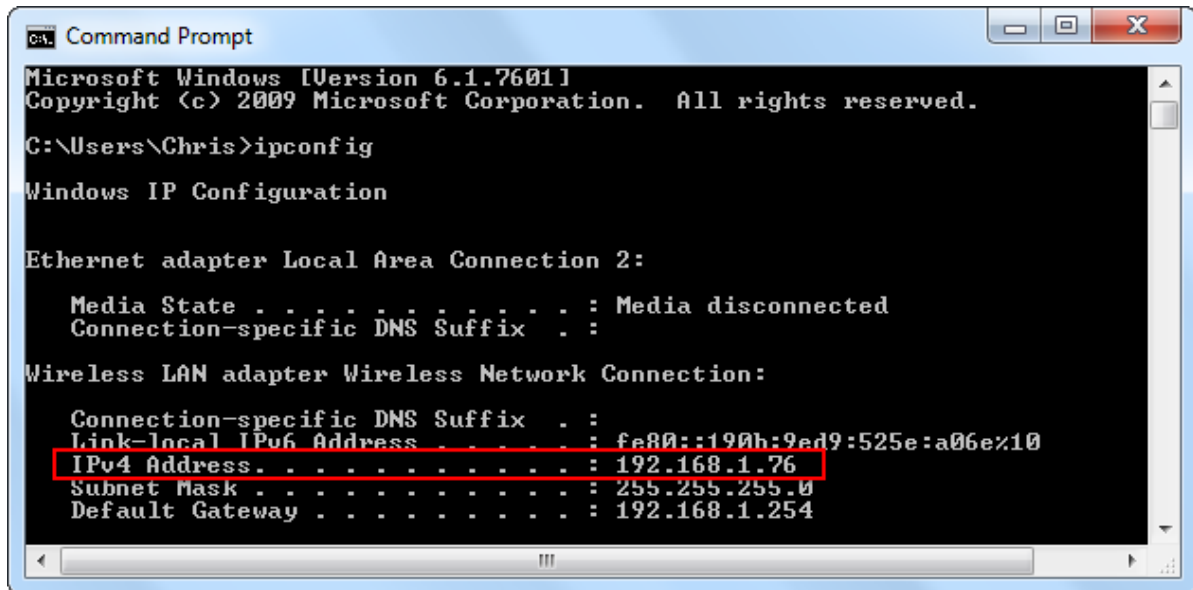
In the **second sample case** the students can fit on the corners of the room, keeping a seat free between them.



## IP Address (ip)

Marco is trying to hack the network in his university. In order to do so, he needs to send a malicious packet to each computer that might be connected to the same network.

Just to be sure, Marco decided to send the packet to a range of *IP addresses*. An IP address is a set of four “octets” separated by dots (each “octet” is an integer in the 0 to 255 inclusive range). For example, 127.0.0.1, 192.168.2.3 and 255.255.255.255 are valid IP addresses, while something like 10.20.300.4000 is not.



```

C:\> Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Chris>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection 2:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

Wireless LAN adapter Wireless Network Connection:

    Connection-specific DNS Suffix  . :
    Link-local IPv6 Address . . . . . : fe80::190b:9ed9:525e:a06e%10
    IPv4 Address. . . . . : 192.168.1.76
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.1.254
  
```

Figure 1: Using the ipconfig command on Windows to obtain the IP.

Given an IP address, we can *increment* it and go to the next address in this way: if the last octet is less than 255 then we increment it by 1, otherwise we set it to 0 and increment the second-last octet (repeating this in a similar way to the left if necessary). This is similar to what happens when we increment a number by 1: if the last digit is 9 then we need to set it to 0 and increment the second-last digit (which could be 9 as well and so on).

For example, if Marco wanted to send the malicious packet to all IPs in the range that goes from 192.168.1.0 to 192.168.1.255, he would need to send a total of 256 packets. If the range was 192.168.1.0 to 192.168.2.0, instead, he would have to send 257 packets.

Calculate, given a valid range of addresses, how many packets Marco needs to send.

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

## Input

The first line contains the first IP address. The second line contains the second IP address.

## Output





You need to write a single line with an integer: the number of packets that Marco should send.

## Constraints

- Both addresses in input are valid IP addresses.
- The first address is lower than or equal to the second.

## 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)      The IP endpoints only differ in the last octet (e.g. 192.168.0.21 to 192.168.0.147)  

- **Subtask 3** (50 points)      The given IP range spans at most 1 000 000 addresses.  

- **Subtask 4** (30 points)      No additional limitations.  


## Examples

input	output
192.168.1.0 192.168.1.255	256
192.168.1.0 192.168.2.0	257
0.1.2.3 200.201.202.203	3368601801



## Maintenance Planning (machine)

At *Amedoro-7-9*, the manufacturing company owned by the problemist Luca, a lot of machines are used for producing all sorts of things. The end of the year is approaching and his accounting department started analyzing the profits and the expenses of the year. The results? Terrible! Luca has become angrier than ever, threatening to fire his employees if they don't find a solution immediately!



Figure 1: *The Machine* which makes things.

Night after night, the employees finally found out the cause of the loss: bad management of *The Machine*. *The Machine* is the main component of the factory, it costs  $C$  euros to be bought new, and every year you'll have to pay  $M[a]$  euros for the maintenance ( $a > 0$  is the age in years of *The Machine*). Of course you can sell an old *The Machine*, recovering part of its costs: after  $a > 0$  years *The Machine* can be sold for  $P[a]$  euros. After  $D$  years *The Machine* is too old to be used and has to be sold.

For example if *The Machine* costs  $C$  and has a lifespan of  $D$  years, after the first year you have to pay  $C + M[1]$ . If you want to sell *The Machine* after 3 years you have to pay a total of  $C + M[1] + M[2] + M[3] - P[3]$ . *The Machine* has to be sold not after the  $D$ -th year.

Help Luca's employees design a cost-saving plan for the future. Knowing that at year 0 you have to buy the machine new, compute the lowest total cost for the following  $Y$  years. Notice that the factory needs to have exactly one *The Machine* in use: you cannot sell *The Machine* and not buy a new one immediately after (or you won't be able to operate), or buy *The Machine* without selling an old one before (or you won't have space to fit both).

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

## Input

The first line contains the integers  $C$ ,  $D$  and  $Y$ , the cost of a new *The Machine*, the maximum lifespan of it and the total number of years to plan.

The second line contains  $D$  integers  $M[a]$ , the cost of the maintenance after  $a$  years of life.

The third line contains  $D$  integers  $P[a]$ , the revenue of selling *The Machine* after  $a$  years of life.

## Output







You need to write a single line with an integer: the total minimum cost of an  $Y$  years plan.

## Constraints

- $1 \leq C \leq 1000$ .
- $1 \leq D \leq 1000$ .
- $1 \leq Y \leq 1\,000\,000$ .
- $0 \leq M[a] \leq 1000$  for  $a = 1 \dots D$ .
- $0 \leq P[a] \leq 1000$  for  $a = 1 \dots D$ .
- It's not guaranteed that  $M$  nor  $P$  are monotonic.

## 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)       $D, Y \leq 10$ .  

- Subtask 3 (25 points)       $M[a] \geq M[a - 1]$  and  $P[a] \leq P[a - 1]$  for all  $a = 2 \dots D$ .  $Y \leq 10\,000$ .  

- Subtask 4 (30 points)       $D, Y \leq 100$ .  

- Subtask 5 (10 points)       $Y \leq 10\,000$ .  

- Subtask 6 (15 points)      No additional limitations.  


## Examples

input	output
10 5 6 1 2 2 5 2 5 4 3 5 4	24
100 3 1000 123 456 789 987 654 321	-764000

## Explanation

In the **first sample case** the best strategy is to:

- Buy the machine at the beginning (pay 10), keep it for 5 years (pay a total of 12 for the maintenance) and sell it after the fifth year (recover 4).
- Buy the machine after the fifth year (pay 10), pay the maintenance for an year (pay 1) and sell it after one year (recover 5).

In the **second sample case** it's possible to sell the machine at a price higher than the buying one, obtaining a positive net profit (hence a negative total cost).

## Encrypted Contacts (ransomware)

Marco's phone has been attacked by hackers with a ransomware: they have remotely encrypted the phone book with all his contacts and now he has been asked to pay a ransom in bitcoins to recover his data.

He has decided to have some fun trying to perform a full reverse engineering of the malware to recover data without having to pay money. Fortunately for him, the encryption scheme is not too sophisticated. The malware encrypts each digit in isolation and substitutes it with a *code*. For every digit from 0 to 9, Marco has been able to determine which was the corresponding code.

For instance, suppose that digit 0 has been replaced with the code 12345 and that the digit 1 has been replaced with the code 1235; in this scenario an hypothetical number 010 would have been encrypted with the sequence of digits 12345123512345.

After all this grueling work, Marco asks you a small help: recover the original unencrypted numbers for his  $N$  contacts in the phone book.

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

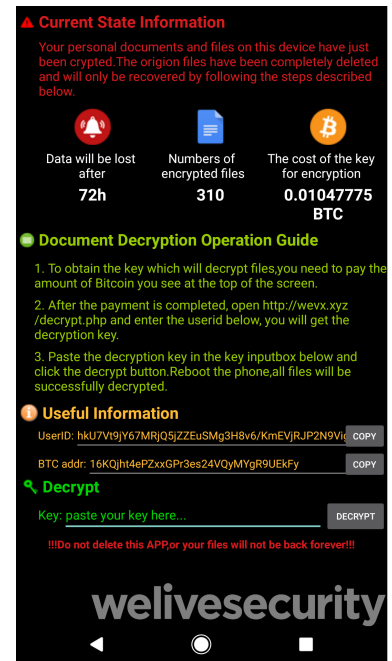


Figure 1: An Android phone infected by ransomware (source: welivesecurity.com)

### Input

The first line contains the number of contacts  $N$ . The following  $N$  lines contain each a string of digits representing the encrypted number. The  $i$ -th of the following (and last) 10 lines contains the encrypted code used to replace the digit  $i$ .

### Output

You need to write  $N$  lines where the  $i$ -th contains the unencrypted number of the  $i$ -th contact.

### Constraints




- $1 \leq N \leq 100$ .
- Each number has at most 1000 digits in its encrypted version.
- Each code used to encrypt a digit is at most 100 digits long.
- It is guaranteed that a solution exists and is unique.

### 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** (30 points)      All codes have length 1.  

- **Subtask 3** (35 points)      All codes have the same length.  

- **Subtask 4** (35 points)      No additional limitations.  


## Examples

input	output
2 333333000111777000 333333000111777111 000 111 222 333 444 555 666 777 888 999	330170 330171
1 717171500500211150050089 500 3 21 71 501 11 0 98 89 42	3330025008

## Explanation

In the **first sample case** all codewords have the same length, three digits. The first contact can be decrypted as 333 (encrypted code for digit 3), 333 (encrypted code for digit 3), 000 (encrypted code for digit 0), 111 (encrypted code for digit 1), 777 (encrypted code for digit 7), 000 (encrypted code for digit 0). The second contact follows the same reasoning.

In the **second sample case** the contact can be decrypted as: 71 (encrypted code for digit 3) three times, 500 (encrypted code for digit 0) twice, 21 (encrypted code for digit 2), 11 (encrypted code for digit 5), 500 (encrypted code for digit 0), and 89 (encrypted code for digit 8).