

Playing with barrels (barrels)

For Christmas William will give to Giorgio a really nice present. Giorgio has always loved playing with barrels and his favorite game consists in taking N barrels, each of which has capacity C_i , and aligning them in a row in a way that when the i -th barrel gets full the water instantly goes to the $(i+1)$ -th barrel.



Figure 1: Some barrels of different size.

After setting up the barrels Giorgio will do either:

- Put k liters of water in the u -th barrel;
- Measure how many liters of water are there in the q -th barrel.

Remember that barrels love overflowing: when some water flows in a full barrel it is all transferred to the next barrel. The last barrel overflows to the floor... but that's not our problem!

Of course, William's present will be a fancy new simulator, not the real thing: help him writing it.

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

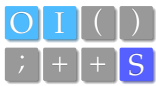
Input

The first line contains two integers N and T . The second line contains N integers, C_i . The next T lines will contain Giorgio's actions, in one of the following formats:

- P k u , put k liters into the u -th barrel (with $0 \leq u < N$).
- M q , measure how many liters are there in the q -th barrel (with $0 \leq q < N$).

Output

For every measure action, write a line with the number of liters in the corresponding barrel.



Constraints

- $1 \leq N, T \leq 1\,000\,000$.
- $1 \leq C_i \leq 10^9$ for each $i = 0 \dots N - 1$.
- $1 \leq k \leq 10^9$ and $0 \leq u, q < N$ for each action.

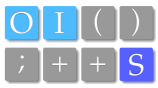
Scoring

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]:** There are no “put” actions.
- **Subtask 3 [20 points]:** $u = 0, q = N - 1$ for every action.
- **Subtask 4 [25 points]:** $u = 0$ for every action.
- **Subtask 5 [30 points]:** $N, T \leq 1000$.
- **Subtask 6 [15 points]:** No additional limitations.

Examples

input.txt	output.txt
4 5 2 2 3 1 P 4 1 P 1 0 M 2 P 5 2 M 3	2 1
3 4 2 2 2 P 1 2 P 3 1 M 0 M 2	0 2
4 3 3 3 1 1 P 4 0 P 3 0 M 2	1



Explanation

In the **first sample case**, 4 liters of water are put in barrel 1, filling it and overflowing 2 liters into barrel 2. Then, a liter is put in the first barrel so that the overall barrels' state becomes 1 2 2 0. Finally, 5 liters are put in barrel 2: two of them fill that barrel, one flows into the last one and the remaining two go to the floor. The final state of the barrels is thus 1 2 3 1.

In the **second sample case**, the final barrels' state is 0 2 2.

In the **third sample case**, the final barrels' state is 3 3 1 0.

Numerological prediction (bloomsday)

Everybody knows that the world should end with the coming of *doomsday*, when all ills will be unleashed, throwing the earth in an hell of ice and fire.

What is not nearly as known is that (given the right astrological conditions) a day of heavenly peace and joy should also come!

Predicting this day, called *bloomsday*, is the ultimate quest of the most renowned numerologists, and Giorgio is no exception. After years of excruciating scientific research, he proved that bloomsday will come when the *pattern of god* will finally appear in the *holy sequence*.

For this quest, Giorgio printed out the first few bazillion numbers of the “holy sequence”, which is the infinite sequence of numbers:

$$K \cdot 1^E, K \cdot 2^E, \dots, K \cdot X^E, \dots$$

where K and E are carefully chosen to maximise holiness. Even though the description of the “pattern of god” is too complex to fit into the margin of this page, Giorgio has already found it in the N -th digit of the printout! Thus, this digit belongs to the X -th element $K \cdot X^E$ where X is the advent of bloomsday, measured in days elapsed from the beginning of the universe.

Sadly, due to his limited research budget, Giorgio had to print the numbers with the lowest quality and smallest font, making impossible to recognise the spaces between subsequent numbers of the sequence. Thus, Giorgio has no clue on which X corresponds to the N he found. Help him find when bloomsday will come, by computing such X !



Figure 1: Artist’s impression of the advent of bloomsday.

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

Input

The first and only line contains integers K, E, N .

Output

You need to write a single line with an integer: the number X such that the N -th digit of the sequence belongs to $K \cdot X^E$.

Constraints

- $1 \leq E, K \leq 10$.
- $1 \leq N, X < 10^{18}$.
- $K \cdot X^E < 10^{18}$ except in subtasks where otherwise stated.



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]:** $K = E = 1, X < 10\,000$.
- **Subtask 3 [15 points]:** $K = E = 1$.
- **Subtask 4 [15 points]:** $E = 1$.
- **Subtask 5 [10 points]:** $X < 10\,000$.
- **Subtask 6 [20 points]:** No additional limitations.
- **Subtask 7 [10 points]:** $K \cdot X^E > 10^{18}$.

Examples

input.txt	output.txt
1 1 14	12
3 2 13	7

Explanation

In the **first sample case**, the holy sequence starts as follows (the N -th digit is highlighted in red):

$\overbrace{1}^1 \overbrace{2}^2 \overbrace{3}^3 \overbrace{4}^4 \overbrace{5}^5 \overbrace{6}^6 \overbrace{7}^7 \overbrace{8}^8 \overbrace{9}^9 \overbrace{10}^{10} \overbrace{11}^{11} \overbrace{12}^{12} \overbrace{13}^{13} \overbrace{14}^{14} \overbrace{15}^{15} \overbrace{16}^{16} \overbrace{17}^{17} \overbrace{18}^{18} \overbrace{19}^{19} \overbrace{20}^{20} \overbrace{21}^{21} \dots$

In the **second sample case**, the holy sequence starts as follows (the N -th digit is highlighted in red):

$\overbrace{3}^1 \overbrace{1\ 2}^2 \overbrace{2\ 7}^3 \overbrace{4\ 8}^4 \overbrace{7\ 5}^5 \overbrace{1\ 0\ 8}^6 \overbrace{1\ 4\ 7}^7 \overbrace{1\ 9\ 2}^8 \overbrace{2\ 4\ 3}^9 \dots$

Olympic cake (cake)

The preparations for the big final competition of this year's Olympiads have already begun!

A big *rectangular* cake will be served at the end of the meal, to celebrate the end of the 2018 edition. Many guests are expected to take part in the festivities and, obviously, everyone wants to taste at least a bit of the special cake.



Figure 1: A rectangular cake baked for the 2014 final of the Italian Olympiad.

The baker wants to please as many people as possible, but time is running out: he can only do *at most* N cuts. Each cut is made either vertically or horizontally, for the whole length of the cake.

How many pieces of cake can the baker obtain at most?

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

Input

The first and only line of the input contains a single integer: N .

Output

You need to write a single line with one integer: the number of pieces that can be obtained at most, making no more than N cuts.

Constraints

- $0 \leq N \leq 1\,000\,000\,000$.

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.

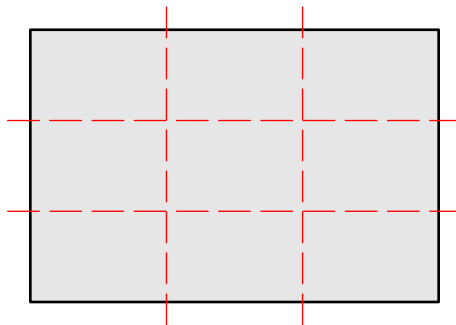
- **Subtask 1** [0 points]: Examples.
- **Subtask 2** [30 points]: $N \leq 1\,000$.
- **Subtask 3** [20 points]: $N \leq 50\,000$.
- **Subtask 4** [20 points]: $N \leq 1\,000\,000$.
- **Subtask 5** [30 points]: No additional limitations.

Examples

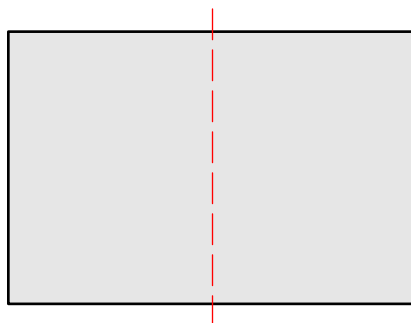
stdin/input.txt	stdout/output.txt
4	9
1	2

Explanation

In the **first example**, an optimal solution is to make two vertical cuts and two horizontal cuts. This splits the cake in nine pieces, as shown in the following picture.



In the **second example**, we can obtain two pieces making a single vertical cut. No other allowed cutting strategy leads to more pieces.



Forgotten attractions (coordonate)

During his *algorithmic excursions* in the Alps, Giorgio likes to stop from time to time and visit some *attractions*. An attraction can be any place: an old abandoned church, a safe house or just a remarkable place from which to watch the stunning landscape. He has marked on the map N of these attractions, each situated at a different altitude of $A[i]$ meters above sea level.

The night before the excursion, Luca wanted to pull a prank on Giorgio and erased all the altitudes from the map. Instead, he left a footnote with a list of $\frac{N \cdot (N-1)}{2}$ numbers where each value is the positive altitude difference between two attractions (reported only once, i.e. for each pair of different attractions (i, j) only the difference $|A[i] - A[j]|$ is listed).

Giorgio has just woken up and found the prank: knowing that among the attractions he also listed his house (located at an altitude of *zero* meters), help him saving his excursion by reconstructing all the altitudes of the attractions!



Figure 1: A safe house in the Alps.

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

Input

The first line contains a single integer N . The second line contains $\frac{N \cdot (N-1)}{2}$ integers D_j , each representing the (absolute) difference between the altitudes of two different attractions (in a random order).

Output

You need to write a single line containing N integers A_i , the altitudes of the attractions in ascending order.

Constraints

- $2 \leq N \leq 1000$.
- $0 \leq A_i \leq 100\,000\,000$ for each $i = 0 \dots N - 1$.
- It is guaranteed that a solution exists.
- Different attractions always have different altitudes: $A_i \neq A_j$ for each pair (i, j) with $i \neq j$.
- When you cannot determine uniquely the altitude of an attraction, you can output *any* sequence of altitudes consistent with the given differences.



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]:** $N \leq 3$.
- **Subtask 3 [30 points]:** $A_i < 20$ for each i .
- **Subtask 4 [25 points]:** $N \leq 8$, $A_i < 100\,000$ for each i .
- **Subtask 5 [20 points]:** $N \leq 100$, $A_i < 100\,000$ for each i .
- **Subtask 6 [10 points]:** No additional limitations.

Examples

stdin/input.txt	stdout/output.txt
4 7 4 6 9 3 2	0 3 7 9
5 2 4 6 2 5 7 9 9 11 3	0 2 6 9 11

Explanation

In the **first sample case**, the possible solutions are 0 2 6 9 and 0 3 7 9.

In the **second sample case**, the possible solutions are 0 2 6 9 11 and 0 2 5 9 11.

Saddest friend (maxim)


Luca is giving a Christmas gift to each of his N friends! When the presents will be opened in the *Giftsgiving Ceremony*, his friends will be aligned in a row, numbered from 0 to $N - 1$, and friend i will receive a gift of well-known value V_i .



Figure 1: Presents getting ready for the Giftsgiving Ceremony.

Luca knows that his friends will be able to see other presents at a distance up to K positions (that is, $K - 1$ people in between). Of course, the friends whose gifts are less valuable will be sad: more precisely, their sadness will be equal to the absolute difference between the maximum value of a present they can see and the value of their one.

Help Luca prepare for the ceremony by computing how sad will be the saddest friend!

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

Input

The first line contains two integers N and K . The second line contains the N integers V_i .

Output

You need to write a single line with an integer: the maximum absolute difference between two values distant at most K .

Constraints

- $2 \leq N \leq 100\,000$.
- $1 \leq K \leq N - 1$
- $1 \leq V_i \leq 30\,000$ 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]:** All the V_i are equal.
- **Subtask 3 [10 points]:** $N \leq 10$.
- **Subtask 4 [30 points]:** $NK \leq 10^6$.
- **Subtask 5 [30 points]:** $V_i \leq V_{i+1}$ for all $i = 0 \dots N - 2$.
- **Subtask 6 [20 points]:** No additional limitations.

Examples

input.txt	output.txt
6 2 5 4 6 5 2 4	4
4 1 1 2 4 5	2

Explanation

In the **first sample case**, the maximum sadness is between the third friend (6) and the fifth one (2).

In the **second sample case**, friends can only see people next to them. Then, the maximum difference is between the second (2) and the third (4).

Apple juice challenge (pouring)

William is a long-time expert in drinking games, and today he invented a fantastic game! He starts with two glasses of different capacities A_C and B_C containing some amount of *apple juice*. The only valid operations are the following:


- Drink **all** the apple juice contained in one of the glasses.
- Refill one of the glasses **completely** (regardless of how full it is already).
- Pour a glass into another **until** either the other glass is full or the first glass is empty.



Figure 1: The game being set up by William with sparkling apple juice.

William chose a “starting configuration” (A_S, B_S) and he is now challenging Giorgio to reach a “goal configuration” (A_G, B_G) , by strictly following the rules of the game.

Giorgio doesn’t want to drink too much apple juice (or spend a lot of time pouring, for that matter): help him reach the goal configuration in the least possible number of steps!

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

Input

The first line contains two integers A_C and B_C , the capacities of the two glasses. The second line contains two integers A_S and B_S , the starting configuration of the game. The third line contains two integers A_G and B_G , the goal configuration of the game.

Output

You need to write a single line with an integer: the minimum number of steps to reach the goal configuration from the starting configuration, by following the rules of the game.

Constraints

- $1 \leq A_C, B_C \leq 10^{18}$.
- $0 \leq A_S, A_G \leq A_C, 0 \leq B_S, B_G \leq B_C$.
- It is guaranteed that a sequence of steps from the starting to the goal configuration exists.

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.

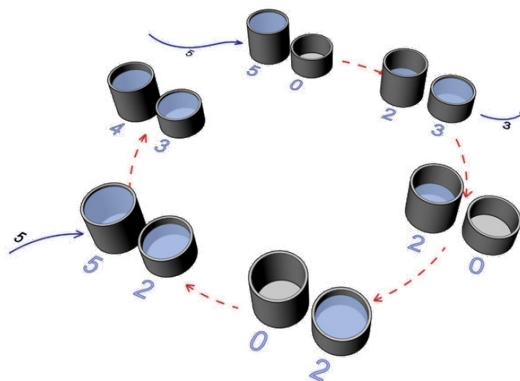
- **Subtask 1 [0 points]:** Examples.
- **Subtask 2 [30 points]:** $A_C, B_C \leq 10$.
- **Subtask 3 [25 points]:** $A_C, B_C \leq 1000$.
- **Subtask 4 [20 points]:** $A_C, B_C \leq 1\,000\,000$.
- **Subtask 5 [15 points]:** $A_S = B_S = 0$.
- **Subtask 6 [10 points]:** No additional limitations.

Examples

input.txt	output.txt
<pre>5 3 0 0 4 3</pre>	6
<pre>1000000000000000000 1000000000000000000 9999999999999999998 1 9999999999999999999 10000000000000000000</pre>	2

Explanation

In the **first sample case**, these are the steps that Giorgio can follow:



In the **second sample case**, you need to pour glass *B* into *A*, and then refill glass *B*.

Gondola network (vapoare)

Santa Claus is coming to town! In Venice, specifically. This city represents a challenge for Santa: in fact, it can be seen as a set of N small islands connected by K navigable canals.

Each canal connects two islands and has a specific width: only a *gondola* which is narrower (or equally wide) than the canal is able to traverse it.

Santa wants to build a network for gift dispatching by renting gondolas in some of the islands, so that every island can be reached by *at least one* gondola. Notice that if an island is not connected to other islands, Santa will still need to rent a gondola for that island: the presents need to be dispatched through the island by boat.



Figure 1: A gondola in Venice.

As anyone that ever visited Venice knows, renting these boats can be quite expensive! You are given the description of every canal (width and islands connected), and a list of all the available gondola widths. For each of these widths, help Santa compute the minimum number of gondolas (of that width) that he would need to rent to build his network!

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

Input

The first line contains three integers: the number of islands N , the number of canals K , the number of available gondola types T . The following K lines contain three integers A_i , B_i , V_i each, describing a navigable canal of width V_i connecting islands A_i and B_i . The following T lines contain one integer W_i each, the width of each gondola type.

Output

You need to write T lines with a single integer each: the minimum number of gondolas that would need to be rented for each available width.

Constraints

- $1 \leq N \leq 50\,000$.
- $1 \leq K \leq 1\,000\,000$.
- $1 \leq T \leq 100\,000$.
- $1 \leq W_i \leq 50\,000$ for each $i = 0 \dots T - 1$.
- Between any two islands there is at most one canal, and all the canals are bidirectional.



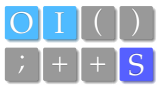
Scoring

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]:** $N \leq 10, K \leq 30, T \leq 20$.
- **Subtask 3 [25 points]:** All values of V_i are equal.
- **Subtask 4 [20 points]:** The graph is a cycle, e.g. with 3 nodes: $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$.
- **Subtask 5 [15 points]:** $T = 1$.
- **Subtask 6 [10 points]:** No additional limitations.

Examples

input.txt	output.txt
6 9 5 1 2 5 1 6 10 2 3 40 2 6 20 3 4 5 3 5 35 3 6 80 4 5 15 5 6 60 20 90 3 60 30	3 6 1 4 3
2 1 5 1 2 3 1 2 3 4 5	1 1 1 2 2



Explanation

In the **first sample case**:

- 3 gondolas of width 20 are necessary: one for the group of lakes (2, 3, 5, 6) and one for each of the lakes 1 and 4.
- A gondola of width 90 cannot sail on any canal, so a ship must be rented for every island.
- A gondola of width 3 can sail on all the canals, so one is enough.
- 4 gondolas of width 60 are necessary: one for the group of lakes (3, 5, 6) and one for each of the lakes 1, 2 and 4.
- 3 gondolas of width 20 are necessary: one for the group of lakes (2, 3, 5, 6) and one for each of the lakes 1 and 4.

In the **second sample case**, one gondola for each island is needed if the width is 4 or above.