The 2021 Sichuan Provincial Collegiate Programming Contest

Contest Session

May 30, 2021

Problem List

A	Chuanpai
B	Hotpot
C	Triangle Pendant
D	Rock Paper Scissors
E	Don't Really Like How The Story Ends
F	Direction Setting
G	Hourly Coding Problem
H	Nihongo wa Muzukashii Desu
I	Monster Hunter
J	Ants
K	K-skip Permutation
L	Spicy Restaurant
M	True Story

This problem set should contain 13 (thirteen) problems on 15 (fifteen) numbered pages. Please inform a runner immediately if something is missing from your problem set.

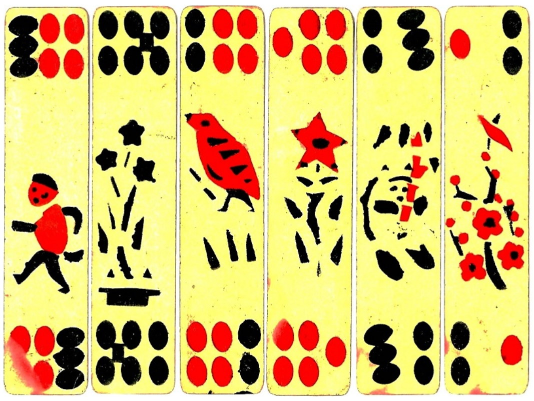
Prepared by SUA Contest Problem Setter Team.
https://sua.ac/

Hosted by

Problem Set Prepared by

Problem A. Chuanpai

Chuanpai(川牌) is a kind of traditional playing cards in Sichuan. Each card is marked with two integers x and y where $1 \le x \le y \le 6$.



Some samples of Chuanpai.
The first one is marked with 3 *and* 4*, while the second one is marked with* 2 *and* 5*.*

Given an integer k, please count the number of different types of cards satisfying $x + y = k$.

We say two cards with integers x_1, y_1 and x_2, y_2 are of different types if $x_1 \neq y_1$ or $x_2 \neq y_2$.

Input

There are multiple test cases. The first line of the input contains an integer T $(1 \le T \le 100)$ indicating the number of test cases. For each test case:

The first and only line contains an integer k $(1 \le k \le 100)$.

Output

For each test case output one line containing one integer, indicating the number of types of cards satisfying $x + y = k$.

Example

standard input	standard output
4 4 5 8 100	2 2 3 0

Note

We use (a, b) to indicate a type of card whose $x = a$ and $y = b$.

For the first sample test case the valid types of cards are $(1, 3)$ and $(2, 2)$.

For the second sample test case the valid types of cards are $(1, 4)$ and $(2, 3)$.

For the third sample test case the valid types of cards are $(2, 6)$, $(3, 5)$ and $(4, 4)$.

Problem B. Hotpot

Sichuan hotpot is one of the most famous dishes around the world. People love its spicy taste.

There are n tourists, numbered from 0 to $(n-1)$, sitting around a hotpot. There are k types of ingredients for the hotpot in total and the i-th tourist favors ingredient a_i most. Initially, every tourist has a happiness value of 0 and the pot is empty.

The tourists will perform m moves one after another, where the i-th (numbered from 0 to $(m-1)$) move is performed by tourist $(i \mod n)$. When tourist t moves:

- If ingredient a_t exists in the pot, he will eat them all and gain 1 happiness value.
- Otherwise, he will put one unit of ingredient a_t into the pot. His happiness value remains unchanged.

Your task is to calculate the happiness value for each tourist after m moves.

Input

There are multiple test cases. The first line of the input contains an integer T $(1 \le T \le 10^3)$ indicating the number of test cases. For each test case:

The first line contains three integers n, k and m $(1 \le n \le 10^5, 1 \le k \le 10^5, 1 \le m \le 10^9)$ indicating the number of tourists, the number of types of ingredients and the number of moves.

The second line contains n integers $a_0, a_1, \cdots, a_{n-1}$ $(1 \le a_i \le k)$ where a_i indicates the favorite ingredient of tourist i.

It's guaranteed that neither the sum of n nor the sum of k of all the test cases will exceed 2×10^5.

Output

For each test case output n integers $h_0, h_1, \cdots, h_{n-1}$ in one line separated by a space, where h_i indicates the happiness value of tourist i after m moves.

Please, DO NOT output extra spaces at the end of each line, or your answer might be considered incorrect!

Example

standard input	standard output

4
3 2 6
1 1 2
1 1 5
1
2 2 10
1 2
2 2 10
1 1
	0 2 1
	2
	2 2
	0 5

Note

The first sample test case is explained as follows:

Move	Tourist	Action	Pot after move
0	0	Puts ingredient 1 into the pot	$\{1\}$
1	1	Eats ingredient 1 in the pot	$\{\}$
2	2	Puts ingredient 2 into the pot	$\{2\}$
3	0	Puts ingredient 1 into the pot	$\{1, 2\}$
4	1	Eats ingredient 1 in the pot	$\{2\}$
5	2	Eats ingredient 2 in the pot	$\{\}$

Problem C. Triangle Pendant

Given a point D at height zero and a triangle $\triangle ABC$ with uniform mass, we use three ropes with length x, y, and z to connect AD, BD, and CD respectively. The mass of the ropes can be ignored. Let the triangle fall naturally and stabilize at the lowest position of the center of gravity. Find the final heights of points A, B, and C.

Input

There are multiple test cases. The first line of the input contains an integer T (about 10^4) indicating the number of test cases. For each test case:

The first and only line contains six integers x, y, z, a, b and c ($1 \le x, y, z, a, b, c \le 1000$, $a + b > c$, $a + c > b$, $b + c > a$) indicating the length of three ropes and the length of BC, AC and AB.

You can assume that the solution always exists.

Output

For each test case output one line containing threes real numbers indicating the height of points A, B and C.

Your answer will be considered correct if and only if the absolute or relative error does not exceed 10^{-4}.

Example

standard input

```
2
1 1 1 1 1 1
2 3 3 1 1 1
```

standard output

```
-0.816496580927726 -0.816496580927726 -0.816496580927726
-2.000000000000000 -2.866025403784439 -2.866025403784439
```

Problem D. Rock Paper Scissors

BaoBao and DreamGrid are playing a card game. Each player has n cards in the beginning and there are three types of cards: rock, paper, and scissors.

The game consists of n rounds. In each round, BaoBao will first play one of his remaining cards (this card is shown to both players). After that, DreamGrid can choose one of his remaining cards and play it (also shown to both players). The score of this round is calculated by referring to the following table:

DreamGrid ↓ BaoBao →	**Rock**	**Paper**	**Scissors**
Rock	0	-1	1
Paper	1	0	-1
Scissors	-1	1	0

After the round, the two played cards are removed from the game. The score of the whole game is the sum of the score of each round.

BaoBao aims at minimizing the score of the whole game, while DreamGrid aims at maximizing it. Both players know the number of cards of each type his opponent and himself holds in the beginning. What's the final score of the game given that both of them take the best strategy?

Input

There are multiple test cases. The first line of the input contains an integer T $(1 \le T \le 10^3)$ indicating the number of test cases. For each test case:

The first line contains three integers b_r, b_p and b_s $(0 \le b_r, b_p, b_s \le 10^9)$, indicating the number of rock, paper and scissors cards BaoBao has.

The second line contains three integers d_r, d_p and d_s $(0 \le d_r, d_p, d_s \le 10^9)$, indicating the number of rock, paper and scissors cards DreamGrid has.

It's guaranteed that $b_r + b_p + b_s = d_r + d_p + d_s$.

Output

For each test case output one line containing one integer indicating the final score of game.

Example

standard input	standard output

4
4 4 2
10 0 0
0 10 0
2 4 4
1 2 3
3 2 1
10 10 10
10 10 10
	-2
	2
	5
	30

Problem E. Don't Really Like How The Story Ends

There are n planets in the galaxy, and many undirected warp tunnels connecting them. 6000 years ago, Spinel performed a depth-first search on the planets, visited all of them, and labeled them from 1 to n in the order of discovery.

Many warp tunnels have broken down since, and only m of them are still working. Spinel wants to know how many new warp tunnels have to be built so that it is possible to perform a depth-first search, where the order of discovery is exactly as labeled 6000 years ago.

Recall that the depth-first search (DFS) algorithm inputs a graph G and a vertex v of G, and labels all vertices reachable from v as discovered.

Here is the pseudocode of a recursive implementation of DFS:

```
procedure DFS(G, v) is
    label v as discovered
    for all vertices w that there exists an edge between v and w do
        if vertex w is not labeled as discovered then
            recursively call DFS(G, w)
```

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains two integers n and m ($1 \le n, m \le 10^5$) indicating the number of planets and the number of remaining warp tunnels.

For the following m lines, the i-th line contains two integers u_i and v_i ($1 \le u_i, v_i \le n$) indicating a warp tunnel between u_i and v_i.

It's guaranteed that the sum of $(n + m)$ of all test cases will not exceed 10^6.

Output

For each test case output one line containing one integer, indicating the minimum number of new warp tunnels that have to be built.

Example

standard input	standard output

3
2 3
1 1
1 2
2 1
4 1
1 4
4 2
1 2
3 4
	0
	2
	1

Note

For the second sample test case we can add a tunnel between planet 1 and 2, and add another tunnel between planet 2 and 3.

For the third sample test case we can add a tunnel between planet 2 and 3.

Problem F. Direction Setting

Given an undirected graph with n vertices and m edges where the i-th vertex has a limit a_i, please assign a direction for each edge so that the graph becomes directed and the following value D is minimized.

$$D = \sum_{i=1}^{n} \max(0, d_i - a_i)$$

where d_i is the in-degree (that is, the number of edges going into that vertex) of the i-th vertex.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains two integers n and m ($2 \le n \le 300$, $1 \le m \le 300$) indicating the number of vertices and edges.

The second line contains n integers $a_1, a_2, \cdots, a_n$ ($0 \le a_i \le 10^4$) where a_i indicates the limit of the i-th vertex.

For the following m lines, the i-th line contains two integers u_i and v_i ($1 \le u_i, v_i \le n$) indicating that there is an edge connecting vertex u_i and v_i. Note that there might be self loops or multiple edges.

It's guaranteed that neither the sum of n nor the sum of m of all test cases will exceed 3×10^3.

Output

For each test case output two lines. The first line contains an integer indicating the smallest possible D. The second line contains a string $s_1 s_2 \cdots s_m$ of length m consisting only of '0's and '1's indicating a direction assignment plan of the edges to achieve the smallest possible D. If s_i = '0' then the i-th edge is going from u_i into v_i; Otherwise it's going from v_i into u_i. If there are multiple valid answers you can output any of them.

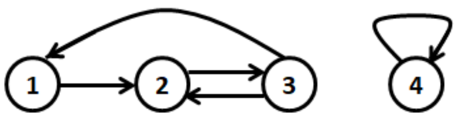
Example

standard input	standard output

2
4 5
0 1 1 5
1 2
1 3
2 3
3 2
4 4
3 2
0 0 2
1 3
3 2
	2
	01001
	0
	01

Note

The first sample test case is shown as follows.



Problem G. Hourly Coding Problem

This problem was asked by Ema.

Given an array of numbers N and an integer k, your task is to split N into k non-empty consecutive partitions such that the maximum sum of any partition is minimized. Output the length of each of the k partitions. If there are multiple solutions, output the solution with the largest lexicographical order.

Solution A is considered to be lexicographically larger than Solution B if there exists an integer $i(1 \le i \le n)$, where the first $i-1$ partitions in A and B have the same length, and the ith partition in A is longer than that in B.

For example, given N = [5, 1, 0, 2, 7, -3, 4] and k = 3, you should output [3, 3, 1], since the optimal partition is [5, 1, 0], [2, 7, -3], [4]. Note that this example used this format solely for convenience, your program should follow the format described below.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains two integers n and k $(1 \le k \le n \le 3 \times 10^5)$ indicating the length of the array and the number of partitions.

The next line contains n integer $N = [a_1, a_2, \cdots, a_n]$ $(|a_i| \le 10^9)$ indicating the array.

It's guaranteed that the sum of n of all test cases will not exceed 6×10^5.

Output

For each test case, output one line containing k integers indicating the length of each of the k partitions. Note again that your answer must be the lexicographically largest answer.

Example

standard input	standard output

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

Problem H. Nihongo wa Muzukashii Desu

Japanese is one of the most difficult languages to learn in the world. Among all those twisted grammar rules, the most troublesome ones for the beginners must be the *verb conjugation* rules.

Japanese verbs appear in different forms under different contexts. By the conjugation rules between their different forms, Japanese verbs can be roughly grouped into three types. We now introduce you the *masu form* to *te form* conjugation rule for the first type of verbs.

- We say a verb is in *masu form* if it ends with “masu” (ます). For example, “naraimasu” (習います, learn) and “nomimasu” (飲みます, drink) are all masu form verbs.
- We say a verb is in *te form* if it ends with “te” (て) or “de” (で). For example, “naratte” (習って, learn) and “nonde” (飲んで, drink) are all te form verbs.
- If the masu form of a verb ends with “imasu” (います), “chimasu” (ちます) or “rimasu” (ります), to change it into its te form, we remove the “imasu”, “chimasu” or “rimasu” at the end and append “tte” (って) to it. For example, “kaimasu” (買います, buy) → “katte” (買って), “machimasu” (待ちます, wait) → “matte” (待って) and “kaerimasu” (帰ります, return) → “kaette” (帰って).
- If the masu form of a verb ends with “mimasu” (みます), “bimasu” (びます) or “nimasu” (にます), to change it into its te form, we remove the “mimasu”, “bimasu” or “nimasu” at the end and append “nde” (んで) to it. For example, “nomimasu” (飲みます, drink) → “nonde” (飲んで), “yobimasu” (呼びます, call) → “yonde” (呼んで) and “shinimasu” (死にます, die) → “shinde” (死んで).
- If the masu form of a verb ends with “kimasu” (きます), to change it into its te form, we remove the “kimasu” at the end append “ite” (いて) to it. For example, “kakimasu” (書きます, write) → “kaite” (書いて). <u>But there is only one verb this rule does not apply, which is the verb “ikimasu” (行きます, go) → “itte” (行って)</u>.
- If the masu form of a verb ends with “gimasu” (ぎます), to change it into its te form, we remove the “gimasu” at the end and append “ide” (いで) to it. For example, “isogimasu” (急ぎます, hurry) → “isoide” (急いで).
- If the masu form of a verb ends with “shimasu” (します), to change it into its te form, we remove the “shimasu” at the end and append “shite” (して) to it. For example, “kashimasu” (貸します, lend) → “kashite” (貸して).

It’s time to check how much you’ve learnt in this lesson! Given a Japanese verb of the first type in its masu form represented in romaji (which means in lower-cased English letters), please change it into its te form.

You might have noticed that if we represent a Japanese verb in romaji, for example “nomimasu”, it’s hard to tell whether this verb ends with “imasu” or “mimasu” for the beginners (actually it ends with “mimasu” as “mi” is one syllable). To simplify this problem, we will not provide you with verbs ending with “imasu”.

Input

There are multiple test cases. The first line of the input contains an integer T (about 100) indicating the number of test cases. For each test case:

The first and only line contains a string s ($1 \le |s| \le 30$) which is a Japanese verb of the first type in its masu form presented in romaji. This verb is guaranteed to end with “chimasu”, “rimasu”, “mimasu”, “bimasu”, “nimasu”, “kimasu”, “gimasu” or “shimasu”.

Output

For each test case output one line containing one string indicating the te form of the verb in romaji.

Example

standard input	standard output
10 machimasu kaerimasu nomimasu yobimasu shinimasu kakimasu ikimasu kikimasu isogimasu kashimasu	matte kaette nonde yonde shinde kaite itte kiite isoide kashite

Problem I. Monster Hunter

Ema is the best carry player in a game. In the game, she needs to eliminate m monsters. The i-th monster has h_i health points (HP) at the beginning. When a monster is attacked by Ema, its HP is reduced by her attack power. When the HP of a monster is less than or equal to 0, the monster is eliminated.

To make the game more interesting, the attack power is not a constant number. There is a basic attack sequence $a_1, a_2, \cdots, a_n$, and the damage caused is generated by repeating this sequence. Formally, let r_i be the damage caused by the i-th attack, we have

$$r_i = \begin{cases} a_i & 1 \le i \le n \\ r_{i-n} & i > n \end{cases}$$

To eliminate the monsters as soon as possible, Ema wants to minimize the number of attacks. Can you tell her the minimum number of attacks required to eliminate all the monsters?

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains an integer n ($1 \le n \le 10^5$) indicating the length of the basic attack sequence.

The second line contains n integers $a_1, a_2, \cdots, a_n$ ($1 \le a_i \le 3$) indicating the basic attack sequence.

The third line contains an integer m ($1 \le m \le 10^5$) indicating the number of monsters.

The fourth line contains m integers $h_1, h_2, \cdots, h_m$ ($1 \le h_i \le 10^9$) where h_i indicates the initial HP of the i-th monster.

It's guaranteed that neither the sum of n nor the sum of m of all test cases will exceed 10^5.

Output

For each test case output one line containing one integer indicating the minimum number of attacks to eliminate all the monsters.

Example

standard input	standard output

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

Note

For the first example, the damage sequence is $3, 2, 3, 2, 3, 2, \cdots$. We can attack monsters 1, 2, 3 and 2 in order to eliminate all the 3 monsters.

For the second example, we can attack monsters 2, 2, 1 in order.

Problem J. Ants

There are n ants living on a stick of length $(10^9 + 1)$ units. The initial position of the i-th ant is a_i units away from the left side of the stick. Some of the ants are facing left at the beginning, while the others are facing right. All ants will move at a speed of 1 unit per second in the direction they're facing. When two ants meet face to face at the same point, both of them will turn around instantly and move on.

There are also two obstacles on the sides of the stick, one located on the leftmost and the other on the rightmost. When an ant runs into one of them, it will also turn around instantly and move on. However, the obstacles aren't indestructible. The left one will break after a hits, while the right one will break for b hits. After an ant passes through a broken obstacle it will fall from the stick. Note that the number of hits is calculated independently for each obstacle, and that the ant which breaks the obstacle will also turn around and will not fall immediately.

In how many seconds will all ants fall from the stick?

Input

There is only one test case in each test file.

The first line of the input contains three integers n, a and b ($1 \le n \le 10^6$, $1 \le a, b \le 10^9$) indicating the number of ants, the number of hits to break the left obstacle and the number of hits to break the right obstacle.

The second line contains n integers $a_1, a_2, \cdots, a_n$ ($1 \le a_i \le 10^9$, $a_i < a_{i+1}$) indicating the initial position of ants.

The third line contains n integers $d_1, d_2, \cdots, d_n$ ($d_i \in \{0, 1\}$). If $d_i = 0$ then the i-th ant is facing left initially, otherwise it is facing right.

Output

Output one line containing one integer indicating the number of seconds for all ants to fall from the stick.

Example

standard input	standard output

2 2 4
2 3
0 1
	4000000001

Note

The sample test case is explained as follows.

Time	Event	Left Hit	Right Hit
2	Ant 1 hits the left obstacle	1	0
999999998	Ant 2 hits the right obstacle	1	1
1000000000.5	Ant 1 meets ant 2 at 999999998.5 units from the left	1	1
1000000003	Ant 2 hits the right obstacle	1	2
1999999999	Ant 1 hits the left obstacle	2	2
2000000001.5	Ant 1 meets ant 2 at 2.5 units from the left	2	2
2000000004	Ant 1 falls from the left	2	2
3000000000	Ant 2 hits the right obstacle	2	3
4000000001	Ant 2 falls from the left	2	3

Problem K. K-skip Permutation

For a permutation $P = p_1, p_2, \cdots, p_n$ of n, let $f(P, k)$ be the number of i satisfying $1 \le i < n$ and $p_i + k = p_{i+1}$.

Given two integers n and k, your task is to find a permutation P of n such that $f(P, k)$ is maximized.

Recall that in a permutation of n, each integer from 1 to n (both inclusive) appears exactly once.

Input

There is only one test case in each test file.

The first and only line contains two integers n and k ($1 \le n, k \le 10^6$).

Output

Output one line containing n integers indicating a permutation P of n that maximizes $f(P, k)$. If there are multiple valid answers you can output any of them.

Please, DO NOT output extra spaces at the end of the line, or your answer may be considered incorrect!

Examples

standard input	standard output
`3 1`	`1 2 3`
`7 3`	`2 5 1 4 7 3 6`
`3 7`	`1 3 2`

Problem L. Spicy Restaurant

There are n hotpot restaurants numbered from 1 to n in Chengdu and the i-th restaurant serves hotpots of a certain spicy value w_i. A higher spicy value indicates a hotter taste, while a lower spicy value is more gentle (still need to be very careful, though).

We can consider these n restaurants as nodes on an undirected graph with m edges. Now we have q tourists who want to give the hotpots a try. Given the current positions of the tourists and the maximum spicy value they can bear, your task is to calculate the shortest distance between a tourist and the closest restaurant he can accept.

In this problem we define the distance of a path as the number of edges in the path.

Input

There is only one test case in each test file.

The first line contains three integers n, m and q ($1 \le n, m \le 10^5, 1 \le q \le 5 \times 10^5$) indicating the number of restaurants, the number of edges and the number of tourists.

The second line contains n integers $w_1, w_2, \cdots, w_n$ ($1 \le w_i \le 100$) where w_i indicates the spicy value of the i-th restaurant.

For the following m lines, the i-th line contains two integers u_i and v_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$) indicating an edge connecting restaurant u_i and v_i.

For the following q lines, the i-th line contains two integers p_i and a_i ($1 \le p_i \le n$, $1 \le a_i \le 100$) indicating that the i-th tourist is currently at restaurant p_i and that the maximum spicy value he can accept is a_i.

Output

Output q lines where the i-th line contains one integer indicating the shortest distance between the i-th tourist and the closest restaurant he can accept. If there is no such restaurant, output "-1" instead.

Example

standard input	standard output

4 4 5
5 4 2 3
1 2
2 3
3 4
4 1
1 1
1 2
1 3
1 4
1 5
	-1
	2
	1
	1
	0

Problem M. True Story

Chengdu Shuangliu International Airport is the major international airport serving Chengdu, the capital of Sichuan province, China. It handled 55.9 million passengers in 2019, being among the world's 25 busiest airports in 2019, the fourth-busiest in mainland China, and the busiest in western China.

Ema and her friends are gonna be late for their flight. Now at the beginning of hour 0, they are x km away from the airport, and the boarding time is at the beginning of hour p_0. There are n people in total (Ema herself included), and the i-th one can travel at a speed of s_i km/h. They will have to reach the airport not later than the boarding time to catch the plane.

However, all is not lost. Ema knows the boarding time will be postponed. The boarding time will be postponed k times: The i-th one will be announced at the beginning of hour t_i, and postpone the time later to the beginning of hour p_i. There are still challenges, though, as everyone will only move when he/she can catch the plane in time. That is if the current time before boarding is not enough for him/her to arrive at the airport, he or she will stop moving and just stay at that point; Otherwise, he/she will move on again from where he/she has stopped, or just keep on moving.

Note that every time the boarding time is postponed, everyone will instantly change their action accordingly. Also, everyone only knows the postponement when it is announced, and cannot act on it beforehand.

Please calculate how many people can catch the plane in the end.

Input

There is only one test case in each test file.

The first line contains four integers n, k, x and p_0 ($1 \le n, k \le 10^5$, $1 \le x, p_0 \le 10^9$) indicating the number of people, the number of postponement, the distance from the starting point to the airport and the initial boarding time.

The second line contains n integers $s_1, s_2, \cdots, s_n$ ($1 \le s_i \le 10^9$) indicating the speed of the i-th person.

The third line contains k integers $t_1, t_2, \cdots, t_k$ ($1 \le t_i \le 10^9$, $t_i < t_{i+1}$, $t_i < p_{i-1}$) indicating the hour when the i-th postponement is announced.

The fourth line contains k integers $p_1, p_2, \cdots, p_k$ ($1 \le p_i \le 10^9$, $p_i < p_{i+1}$) indicating the hour the boarding time is postponed to in the i-th announcement.

Output

Output one line containing one integer indicating the number of people that can catch the plane eventually.

Examples

standard input	standard output
4 3 10 4	2
1 5 2 1	
3 4 5	
7 9 10	
1 3 10 3	0
1	
2 3 4	
5 8 10	

Note

For the first sample test case, at the beginning of hour 0 only the person with speed 5km/h starts moving an arrives at the airport at the beginning of hour 2. Then at the beginning of hour 4 the person with speed 2km/h also starts moving an arrives at the airport at the beginning of hour 9. Only these two people can catch the plane and the other two never moves, as none of the three postponements allow them to arrive in time.

For the second sample test case the only person never moves. If he were to start moving from the very beginning he would catch the plane, however he chose to give up. This story tells us that not all efforts result in success, but giving up is sure to result in failure.