A. Omkar and Password

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Lord Omkar has permitted you to enter the Holy Church of Omkar! To test your worthiness, Omkar gives you a password which you must interpret!

A password is an array a of n positive integers. You apply the following operation to the array: pick any two adjacent numbers that are not equal to each other and replace them with their sum. Formally, choose an index i such that $1 \leq i < n$ and $a_i \neq a_{i+1}$, delete both a_i and a_{i+1} from the array and put $a_i + a_{i+1}$ in their place.

For example, for array $[7, 4, 3, 7]$ you can choose $i = 2$ and the array will become $[7, 4+3, 7] = [7, 7, 7]$. Note that in this array you can't apply this operation anymore.

Notice that one operation will decrease the size of the password by 1. What is the shortest possible length of the password after some number (possibly 0) of operations?

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 100$). Description of the test cases follows.

The first line of each test case contains an integer n ($1 \leq n \leq 2 \cdot 10^5$) — the length of the password.

The second line of each test case contains n integers $a_1, a_2, \dots, a_n$ ($1 \leq a_i \leq 10^9$) — the initial contents of your password.

The sum of n over all test cases will not exceed $2 \cdot 10^5$.

Output

For each password, print one integer: the shortest possible length of the password after some number of operations.

Example

input

```
2
4
2 1 3 1
2
420 420
```

output

```
1
2
```

Note

In the first test case, you can do the following to achieve a length of 1:

Pick $i = 2$ to get $[2, 4, 1]$

Pick $i = 1$ to get $[6, 1]$

Pick $i = 1$ to get $[7]$

In the second test case, you can't perform any operations because there is no valid i that satisfies the requirements mentioned above.

B. Omkar and Infinity Clock

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Being stuck at home, Ray became extremely bored. To pass time, he asks Lord Omkar to use his time bending power: Infinity Clock! However, Lord Omkar will only listen to mortals who can solve the following problem:

You are given an array a of n integers. You are also given an integer k. Lord Omkar wants you to do k operations with this array.

Define one operation as the following:

1. Set d to be the maximum value of your array.
2. For every i from 1 to n, replace a_i with $d - a_i$.

The goal is to predict the contents in the array after k operations. Please help Ray determine what the final sequence will look like!

Input

Each test contains multiple test cases. The first line contains the number of cases t ($1 \le t \le 100$). Description of the test cases follows.

The first line of each test case contains two integers n and k ($1 \le n \le 2 \cdot 10^5, 1 \le k \le 10^{18}$) – the length of your array and the number of operations to perform.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ $(-10^9 \le a_i \le 10^9)$ – the initial contents of your array.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each case, print the final version of array a after k operations described above.

Example

input

```
3
2 1
-199 192
5 19
5 -1 4 2 0
1 2
69
```

output

```
391 0
0 6 1 3 5
0
```

Note

In the first test case the array changes as follows:

- Initially, the array is $[-199, 192]$. $d = 192$.
- After the operation, the array becomes $[d - (-199), d - 192] = [391, 0]$.

C. Omkar and Waterslide

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Omkar is building a waterslide in his water park, and he needs your help to ensure that he does it as efficiently as possible.

Omkar currently has n supports arranged in a line, the i-th of which has height a_i. Omkar wants to build his waterslide from the right to the left, so his supports must be nondecreasing in height in order to support the waterslide. In 1 operation, Omkar can do the following: take any **contiguous subsegment** of supports which is **nondecreasing by heights** and add 1 to each of their heights.

Help Omkar find the minimum number of operations he needs to perform to make his supports able to support his waterslide!

An array b is a subsegment of an array c if b can be obtained from c by deletion of several (possibly zero or all) elements from the beginning and several (possibly zero or all) elements from the end.

An array $b_1, b_2, \ldots, b_n$ is called nondecreasing if $b_i \le b_{i+1}$ for every i from 1 to $n - 1$.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 100$). Description of the test cases follows.

The first line of each test case contains an integer n ($1 \le n \le 2 \cdot 10^5$) — the number of supports Omkar has.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ $(0 \le a_i \le 10^9)$ — the heights of the supports.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output a single integer — the minimum number of operations Omkar needs to perform to make his supports able to support his waterslide.

Example

input

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

output

```
3
2
0
```

Note

The subarray with which Omkar performs the operation is bolded.

In the first test case:

- First operation:

  $$[5, 3, \mathbf{2}, 5] \to [5, 3, \mathbf{3}, 5]$$

- Second operation:

  $$[5, \mathbf{3}, \mathbf{3}, 5] \to [5, \mathbf{4}, \mathbf{4}, 5]$$

- Third operation:

  $$[5, \mathbf{4}, \mathbf{4}, 5] \to [5, \mathbf{5}, \mathbf{5}, 5]$$

In the third test case, the array is already nondecreasing, so Omkar does 0 operations.

D. Omkar and Bed Wars

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

Omkar is playing his favorite pixelated video game, Bed Wars! In Bed Wars, there are n players arranged in a circle, so that for all j such that $2 \leq j \leq n$, player $j - 1$ is to the left of the player j, and player j is to the right of player $j - 1$. Additionally, player n is to the left of player 1, and player 1 is to the right of player n.

Currently, each player is attacking either the player to their left or the player to their right. This means that each player is currently being attacked by either 0, 1, or 2 other players. A key element of Bed Wars strategy is that if a player is being attacked by exactly 1 other player, then they should logically attack that player in response. If instead a player is being attacked by 0 or 2 other players, then Bed Wars strategy says that the player can logically attack either of the adjacent players.

Unfortunately, it might be that some players in this game are not following Bed Wars strategy correctly. Omkar is aware of whom each player is currently attacking, and he can talk to any amount of the n players in the game to make them instead attack another player — i. e. if they are currently attacking the player to their left, Omkar can convince them to instead attack the player to their right; if they are currently attacking the player to their right, Omkar can convince them to instead attack the player to their left.

Omkar would like all players to be acting logically. Calculate the minimum amount of players that Omkar needs to talk to so that after all players he talked to (if any) have changed which player they are attacking, all players are acting logically according to Bed Wars strategy.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 10^4$). The descriptions of the test cases follows.

The first line of each test case contains one integer n ($3 \leq n \leq 2 \cdot 10^5$) — the amount of players (and therefore beds) in this game of Bed Wars.

The second line of each test case contains a string s of length n. The j-th character of s is equal to L if the j-th player is attacking the player to their left, and R if the j-th player is attacking the player to their right.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output one integer: the minimum number of players Omkar needs to talk to to make it so that all players are acting logically according to Bed Wars strategy.

It can be proven that it is always possible for Omkar to achieve this under the given constraints.

Example

input

```
5
4
```

```
RLRL
6
LRRRRL
8
RLLRRRLL
12
LLLLRRLRRRLL
5
RRRRR
```

output

```
0
1
1
3
2
```

Note

In the first test case, players 1 and 2 are attacking each other, and players 3 and 4 are attacking each other. Each player is being attacked by exactly 1 other player, and each player is attacking the player that is attacking them, so all players are already being logical according to Bed Wars strategy and Omkar does not need to talk to any of them, making the answer 0.

In the second test case, not every player acts logically: for example, player 3 is attacked only by player 2, but doesn't attack him in response. Omkar can talk to player 3 to convert the attack arrangement to LR**L**RRL, in which you can see that all players are being logical according to Bed Wars strategy, making the answer 1.

E. Omkar and Duck

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is an interactive problem.

Omkar has just come across a duck! The duck is walking on a grid with n rows and n columns ($2 \le n \le 25$) so that the grid contains a total of n^2 cells. Let's denote by (x, y) the cell in the x-th row from the top and the y-th column from the left. Right now, the duck is at the cell $(1, 1)$ (the cell in the top left corner) and would like to reach the cell (n, n) (the cell in the bottom right corner) by moving either down 1 cell or to the right 1 cell each second.

Since Omkar thinks ducks are fun, he wants to play a game with you based on the movement of the duck. First, for each cell (x, y) in the grid, you will tell Omkar a nonnegative integer $a_{x,y}$ not exceeding 10^{16}, and Omkar will then put $a_{x,y}$ uninteresting problems in the cell (x, y). After that, the duck will start their journey from $(1, 1)$ to (n, n). For each cell (x, y) that the duck crosses during their journey (including the cells $(1, 1)$ and (n, n)), the duck will eat the $a_{x,y}$ uninteresting problems in that cell. Once the duck has completed their journey, Omkar will measure their mass to determine the total number k of uninteresting problems that the duck ate on their journey, and then tell you k.

Your challenge, given k, is to exactly reproduce the duck's path, i. e. to tell Omkar precisely which cells the duck crossed on their journey. To be sure of your mastery of this game, Omkar will have the duck complete q different journeys ($1 \le q \le 10^3$). Note that all journeys are independent: at the beginning of each journey, the cell (x, y) will still contain $a_{x,y}$ uninteresting tasks.

Interaction

The interaction will begin with a line containing a single integer n ($2 \le n \le 25$), the amount of rows and columns in the grid. Read it.

Your program should then print n lines. The x-th line should contain n integers $a_{x,1}, a_{x,2}, \dots, a_{x,n}$ satisfying $0 \le a_{x,y} \le 10^{16}$, where $a_{x,y}$ is the amount of uninteresting problems Omkar should place in the cell (x, y).

After that, you will first receive a single integer q, the amount of journeys that the duck will take. q queries will follow; each query will consist of a single line containing an integer k, the amount of uninteresting problems that the duck ate on that journey. After each query, given that you have determined that the duck visited the cells $(x_1, y_1), (x_2, y_2), \dots, (x_{2n-1}, y_{2n-1})$ in that order (it should always be true that $(x_1, y_1) = (1, 1)$ and $(x_{2n-1}, y_{2n-1}) = (n, n)$), you should output $2n - 1$ lines so that the j-th line contains the two integers x_j, y_j.

Bear in mind that if the sum on your path is k, but your path is different from the actual hidden path, then your solution is still wrong!

After printing each line do not forget to output end of line and flush the output. Otherwise, you will get `Idleness limit exceeded`. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see documentation for other languages.

Hack Format

To hack, first output a line containing n and another line containing q. It must be true that $2 \le n \le 25$ and $1 \le q \le 1000$. Then, output the q journeys taken by the duck in the same format as described above: for each journey, given that the duck visited the cells $(x_1, y_1), (x_2, y_2), \ldots, (x_{2n-1}, y_{2n-1})$ in that order, you should output $2n-1$ lines so that the j-th line contains the two integers x_j, y_j. It must be true that $(x_1, y_1) = (1, 1)$ and $(x_{2n-1}, y_{2n-1}) = (n, n)$. Additionally, for each j such that $2 \le j \le 2n-1$, it must be true that $1 \le x_j, y_j \le n$ and either $(x_j, y_j) = (x_{j-1}+1, y_{j-1})$ or $(x_j, y_j) = (x_{j-1}, y_{j-1}+1)$.

Example

input

```
4




3
23







26







27
```

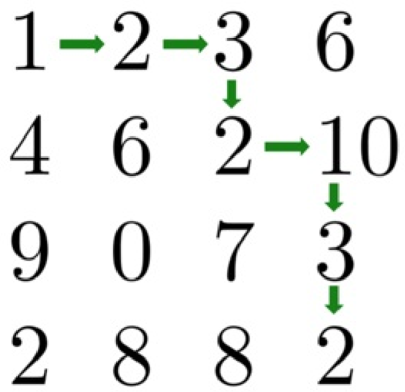
output

```
1 2 3 6
4 6 2 10
9 0 7 3
2 8 8 2


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

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

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

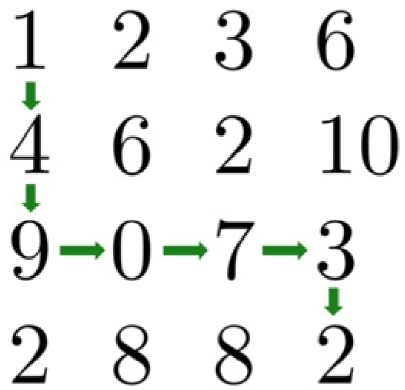
Note

The duck's three journeys are illustrated below.

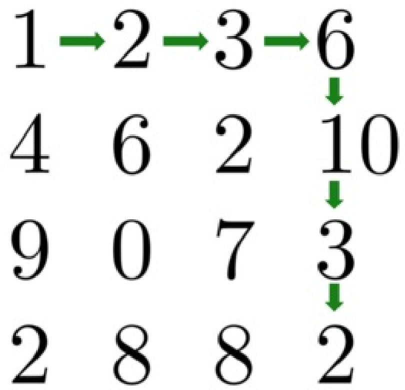
$$1 + 2 + 3 + 2 + 10 + 3 + 2 = 23$$



$1+4+9+0+7+3+2=26$



$1+2+3+6+10+3+2=27$



F. Omkar and Landslide

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Omkar is standing at the foot of Celeste mountain. The summit is n meters away from him, and he can see all of the mountains up to the summit, so for all $1 \le j \le n$ he knows that the height of the mountain at the point j meters away from himself is h_j meters. It turns out that for all j satisfying $1 \le j \le n-1$, $h_j < h_{j+1}$ (meaning that heights are strictly increasing).

Suddenly, a landslide occurs! While the landslide is occurring, the following occurs: every minute, if $h_j + 2 \le h_{j+1}$, then one square meter of dirt will slide from position $j+1$ to position j, so that h_{j+1} is decreased by 1 and h_j is increased by 1. These changes occur simultaneously, so for example, if $h_j + 2 \le h_{j+1}$ and $h_{j+1} + 2 \le h_{j+2}$ for some j, then h_j will be increased by 1, h_{j+2} will be decreased by 1, and h_{j+1} will be both increased and decreased by 1, meaning that in effect h_{j+1} is unchanged during that minute.

The landslide ends when there is no j such that $h_j + 2 \le h_{j+1}$. Help Omkar figure out what the values of $h_1, \dots, h_n$ will be after the landslide ends. It can be proven that under the given constraints, the landslide will always end in finitely many minutes.

Note that because of the large amount of input, it is recommended that your code uses fast IO.

Input

The first line contains a single integer n ($1 \le n \le 10^6$).

The second line contains n integers $h_1, h_2, \dots, h_n$ satisfying $0 \le h_1 < h_2 < \dots < h_n \le 10^{12}$ — the heights.

Output

Output n integers, where the j-th integer is the value of h_j after the landslide has stopped.

Example

input

```
4
2 6 7 8
```

output

```
5 5 6 7
```

Note

Initially, the mountain has heights $2, 6, 7, 8$.

In the first minute, we have $2 + 2 \le 6$, so 2 increases to 3 and 6 decreases to 5, leaving $3, 5, 7, 8$.

In the second minute, we have $3 + 2 \le 5$ and $5 + 2 \le 7$, so 3 increases to 4, 5 is unchanged, and 7 decreases to 6, leaving $4, 5, 6, 8$.

In the third minute, we have $6 + 2 \le 8$, so 6 increases to 7 and 8 decreases to 7, leaving $4, 5, 7, 7$.

In the fourth minute, we have $5 + 2 \le 7$, so 5 increases to 6 and 7 decreases to 6, leaving $4, 6, 6, 7$.

In the fifth minute, we have $4 + 2 \le 6$, so 4 increases to 5 and 6 decreases to 5, leaving $5, 5, 6, 7$.

In the sixth minute, nothing else can change so the landslide stops and our answer is $5, 5, 6, 7$.

G. Omkar and Pies

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Omkar has a pie tray with k $(2 \le k \le 20)$ spots. Each spot in the tray contains either a chocolate pie or a pumpkin pie. However, Omkar does not like the way that the pies are currently arranged, and has another ideal arrangement that he would prefer instead.

To assist Omkar, n elves have gathered in a line to swap the pies in Omkar's tray. The j-th elf from the left is able to swap the pies at positions a_j and b_j in the tray.

In order to get as close to his ideal arrangement as possible, Omkar may choose a contiguous subsegment of the elves and then pass his pie tray through the subsegment starting from the left. However, since the elves have gone to so much effort to gather in a line, they request that Omkar's chosen segment contain at least m $(1 \le m \le n)$ elves.

Formally, Omkar may choose two integers l and r satisfying $1 \le l \le r \le n$ and $r - l + 1 \ge m$ so that first the pies in positions a_l and b_l will be swapped, then the pies in positions a_{l+1} and b_{l+1} will be swapped, etc. until finally the pies in positions a_r and b_r are swapped.

Help Omkar choose a segment of elves such that the amount of positions in Omkar's final arrangement that contain the same type of pie as in his ideal arrangement is the maximum possible. **Note that since Omkar has a big imagination, it might be that the amounts of each type of pie in his original arrangement and in his ideal arrangement do not match**.

Input

The first line contains three integers n, m, and k $(1 \le m \le n \le 10^6$ and $2 \le k \le 20)$ — the number of elves, the minimum subsegment length, and the number of spots in Omkar's tray respectively.

The second and third lines each contain a string of length k consisting of 0s and 1s that represent initial arrangement of pies and ideal arrangement of pies; the j-th character in each string is equal to 0 if the j-th spot in the arrangement contains a chocolate pie and is equal to 1 if the j-th spot in the arrangement contains a pumpkin pie. It is not guaranteed that the two strings have the same amount of 0s or the same amount of 1s.

n lines follow. The j-th of these lines contains two integers a_j and b_j $(1 \le a_j, b_j \le k$, $a_j \neq b_j)$ which indicate that the j-th elf from the left can swap the pies at positions a_j and b_j in the tray.

Output

Output two lines.

The first line should contain a single integer s $(0 \le s \le k)$ equal to the amount of positions that contain the same type of pie in Omkar's final arrangement and in Omkar's ideal arrangement; s should be the maximum possible.

The second line should contain two integers l and r satisfying $1 \le l \le r \le n$ and $r - l + 1 \ge m$, indicating that Omkar should pass his tray through the subsegment $l, l + 1, \dots, r$ to achieve a final arrangement with s positions having the same type of pie as his ideal arrangement.

If there are multiple answers you may output any of them.

Examples

input

```
4 2 5
11000
00011
1 3
3 5
4 2
3 4
```

output

```
5
1 3
```

input

```
4 3 5
11000
00011
1 3
1 5
2 4
1 5
```

output

```
3
1 4
```

Note

In the first test case, the swaps will go like this:

- Swap 1 and 3: `11000` becomes `01100`
- Swap 3 and 5: `01100` becomes `01001`
- Swap 4 and 2: `01001` becomes `00011`

The final arrangement is the same as the ideal arrangement `00011`, so there are 5 positions with the same type of pie, which is optimal.
In the second test case, the swaps will go like this:

- Swap 1 and 3: `11000` becomes `01100`
- Swap 1 and 5: `01100` becomes `01100`
- Swap 4 and 2: `01100` becomes `00110`
- Swap 1 and 5: `00110` becomes `00110`

The final arrangement has 3 positions with the same type of pie as the ideal arrangement `00011`, those being positions 1, 2, and 4. In this case the subsegment of elves $(l, r) = (2, 3)$ is more optimal, but that subsegment is only length 2 and therefore does not satisfy the constraint that the subsegment be of length at least $m = 3$.

H. ZS Shuffles Cards

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

`zscoder` has a deck of $n + m$ custom-made cards, which consists of n cards labelled from 1 to n and m jokers. Since `zscoder` is lonely, he wants to play a game with himself using those cards.

Initially, the deck is shuffled uniformly randomly and placed on the table. `zscoder` has a set S which is initially empty.

Every second, `zscoder` draws the top card from the deck.

- If the card has a number x written on it, `zscoder` removes the card and adds x to the set S.
- If the card drawn is a joker, `zscoder` places all the cards back into the deck and reshuffles (uniformly randomly) the $n + m$ cards to form a new deck (hence the new deck now contains all cards from 1 to n and the m jokers). Then, if S currently contains all the elements from 1 to n, the game ends. Shuffling the deck doesn't take time at all.

What is the expected number of seconds before the game ends? We can show that the answer can be written in the form $\frac{P}{Q}$ where P, Q are relatively prime integers and $Q \neq 0 \bmod 998244353$. Output the value of $(P \cdot Q^{-1})$ modulo 998244353.

Input

The only line of the input contains two integers n and m ($1 \leq n, m \leq 2 \cdot 10^6$).

Output

Output a single integer, the value of $(P \cdot Q^{-1})$ modulo 998244353.

Examples

input

```
2 1
```

output
5

input
3 2
output
332748127

input
14 9
output
969862773

Note

For the first sample, it can be proven that the expected time before the game ends is 5 seconds.

For the second sample, it can be proven that the expected time before the game ends is $\frac{28}{3}$ seconds.

I. Kevin and Grid

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

As Kevin is in BigMan's house, suddenly a trap sends him onto a grid with n rows and m columns.

BigMan's trap is configured by two arrays: an array $a_1, a_2, \ldots, a_n$ and an array $b_1, b_2, \ldots, b_m$.

In the i-th row there is a heater which heats the row by a_i degrees, and in the j-th column there is a heater which heats the column by b_j degrees, so that the temperature of cell (i, j) is $a_i + b_j$.

Fortunately, Kevin has a suit with one parameter x and two modes:

- heat resistance. In this mode suit can stand all temperatures greater or equal to x, but freezes as soon as reaches a cell with temperature less than x.
- cold resistance. In this mode suit can stand all temperatures less than x, but will burn as soon as reaches a cell with temperature at least x.

Once Kevin lands on a cell the suit automatically turns to cold resistance mode if the cell has temperature less than x, or to heat resistance mode otherwise, and cannot change after that.

We say that two cells are adjacent if they share an edge.

Let a path be a sequence $c_1, c_2, \ldots, c_k$ of cells such that c_i and c_{i+1} are adjacent for $1 \leq i \leq k-1$.

We say that two cells are connected if there is a path between the two cells consisting only of cells that Kevin can step on.

A connected component is a maximal set of pairwise connected cells.

We say that a connected component is **good** if Kevin can escape the grid starting from it — when it contains at least one border cell of the grid, and that it's **bad** otherwise.

To evaluate the situation, Kevin gives a score of 1 to each good component and a score of 2 for each bad component.

The final score will be the difference between the total score of components with temperatures bigger than or equal to x and the score of components with temperatures smaller than x.

There are q possible values of x that Kevin can use, and for each of them Kevin wants to know the final score.

Help Kevin defeat BigMan!

Input

The first line contains three integers n,m,q $(1 \leq n, m, q \leq 10^5)$ – the number of rows, columns, and the number of possible values for x respectively.

The second line contains n integers $a_1, a_2, \ldots, a_n$ $(1 \leq a_i \leq 10^5)$.

The third line contains m integers $b_1, b_2, \ldots, b_m$ $(1 \leq b_i \leq 10^5)$.

Each of the next q lines contains one integer x $(1 \leq x \leq 2 \cdot 10^5)$.

Output

Output q lines, in the i-th line output the answer for the i-th possible value of x from the input.

Examples

input

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

output

```
-1
```

input

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

output

```
0
1
```

Note

In the first example, the score for components with temperature smaller than 5 is $1+2$, and the score for components with temperature at least 5 is 2. Thus, the final score is $2-3=-1$.

Codeforces (c) Copyright 2010-2022 Mike Mirzayanov
The only programming contests Web 2.0 platform