Problem A. Another Holiday Activity

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

The holidays are coming, so Helen's family decided to stock up on groceries at the store. Since the family is large and consists of n people, they had to take a lot of groceries and distribute them into bags.

In a family with many people, discipline is essential, so after leaving the store, they lined up and numbered themselves from left to right with numbers from 1 to n. Initially, the i-th family member took a_i bags. It is known that the total number of bags taken by all family members is divisible by n.

To ensure everyone is on equal footing, the family members decided that each of them would carry the same number of bags. To achieve this, every minute, **each** family member can perform one of four actions:

- Do nothing;
- Pass one bag to the family member on the right (if there is a person on the right and the giver has at least one bag);
- Pass one bag to the family member on the left (if there is a person on the left and the giver has at least one bag);
- Pass one bag to the family member on the right and one bag to the family member on the left (if there are people on both sides and the giver has at least two bags).

The transfer of bags between different pairs of neighbors occurs **simultaneously**. For example, in one minute, a certain family member can give one bag to both the left and right; meanwhile, other family members can also transfer bags in that same minute.

Before starting the process of equalizing the number of bags among family members, q changes occur. During the j-th change, the person numbered u_j transfers exactly x_j bags to the person numbered v_j. It is guaranteed that at that moment, person u_j has at least x_j bags.

All changes occur sequentially and are permanent. For example, if during the first change person 1 transferred 2 bags, and in the second change, they transferred 3 bags, then after two changes, they will have 5 bags less than they originally had.

After each change, you need to determine the minimum number of minutes required for the family members to redistribute the bags according to the rules described above so that everyone has the same number of bags.

Input

Each test consists of several test cases. The first line contains a single integer t ($1 \le t \le 100\,000$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains two integers n and q ($2 \le n \le 300\,000$, $1 \le q \le 300\,000$) — the number of family members and the number of bag transfers.

The second line of the test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$) — the initial number of bags each family member has. Let $s = \sum_{i=1}^{n} a_i$. It is guaranteed that s is divisible by n.

Each of the following q lines of the test case contains three integers u_i, v_i, and x_i ($1 \le u_i, v_i \le n$, $1 \le x_i \le s$, $u_i \ne v_i$) — the description of the bag transfer in the format described in the problem statement. It is guaranteed that person u_i has at least x_i bags at the moment of transfer.

It is guaranteed that the sum of n across all test cases does not exceed $300\,000$. It is also guaranteed that the sum of q across all test cases does not exceed $300\,000$.

Output

For each test case, output q integers. The i-th of them should equal the minimum number of minutes required for the family members to redistribute the bags according to the rules described above so that everyone has the same number of bags after the i-th change.

Example

standard input	standard output

2
5 2
1 2 3 4 5
5 1 2
5 4 2
3 1
12 3 9
2 3 3
	1
	2
	4

Note

In the first test case, after the first change, the number of bags with people will be: $3, 2, 3, 4, 3$. To make the number of bags equal for everyone in one minute, the family members can proceed as follows:

- The first person does nothing;
- The second person does nothing;
- The third person passes one bag to the neighbor on the left (the person numbered 2);
- The fourth person passes one bag to the neighbor on the left (the person numbered 3);
- The fifth person does nothing.

Thus, the first person will have 3 bags left, the second will receive 1 bag, the third will give and receive 1 bag, the fourth will give 1 bag, and the fifth will remain with the same number of bags. In total, each person will have exactly 3 bags.

After the second change, the initial number of bags with people will be: $3, 2, 3, 6, 1$. In the first minute, the fourth person will give one bag to both neighbors. After that, the number of bags with people will be: $3, 2, 4, 4, 2$. In the second minute, the third will give a bag to the neighbor on the left, and the fourth will give one to the neighbor on the right. After that, everyone will have 3 bags.

In the second test case, after the only change, the number of bags with people will be: $12, 0, 12$. To make the number of bags equal for everyone, the first person must give a bag to the neighbor on the right for four minutes, and the third person must give a bag to the neighbor on the left. After that, everyone will have 8 bags.

Problem B. Basic Problem from Little S.

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Little S. went on vacation to country $\mathcal{J}$. Country $\mathcal{J}$ is represented as a table of size $n \times m$, with rows numbered by integers from 1 to n and columns numbered by integers from 1 to m. Each cell in the table has its own height above sea level. The height of the cell at coordinates (i, j) is initially equal to $h_{i,j}$.

Little S. plans to stay in country $\mathcal{J}$ for q days, and on the i-th day, one of the following events may occur:

- The authorities of country $\mathcal{J}$ decide to engage in terraforming and change the height of cell (i, j) to x (after this event, it holds that $h_{i,j} = x$);
- Either a low tide or a flood occurs, causing the heights of all cells to increase by x. Note that the heights of some cells may be negative.

It turns out that on any given day, the following *property* holds:

- For any cell (i, j) that *is not on the border of the field*, there exists a neighbor **by side** cell (i', j') such that $h_{i',j'} < h_{i,j}$. A cell with coordinates (i, j) is considered not to be on the border of the field if $2 \le i \le n - 1$ and $2 \le j \le m - 1$.

Little S. noticed that due to these changes, the number of *islands* in country $\mathcal{J}$ may change. An island is defined as any maximal connected subset of cells with **positive** height. In other words, a set of cells is an island if the following conditions are met:

- Every cell in the island has a positive height;
- A path can be constructed between any two cells in the same island using only neighboring cells from that island;
- For any cell not belonging to the island, it holds that it either has a height less than or equal to zero, or it cannot reach any cell of the island by moving only through cells with positive height, transitioning to neighboring cells.

Little S. is very interested in the system of islands in country $\mathcal{J}$, and he wants to know how their number changes. Help Little S. determine how many islands will be in country $\mathcal{J}$ after the first i changes for each i from 1 to q.

Input

Each test consists of several test cases. The first line contains a single integer t ($1 \le t \le 100$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains three integers n, m, and q ($1 \le n, m \le 1\,000$, $1 \le q \le 100\,000$).

Each of the following n lines contains m integers $h_{i,1}, \ldots, h_{i,m}$ ($|h_{ij}| \le 10^9$) — the initial heights of the cells.

Each of the following q lines describes the events of the day in which changes occur. The description corresponds to the format given below:

- `set` i j x ($1 \le i \le n$, $1 \le j \le m$, $|x| \le 10^9$) — the height of cell (i, j) becomes equal to x;
- `water` x ($|x| \le 10^9$) — x is added to the heights of all cells.

It is guaranteed that at any moment in time, the heights of the cells satisfy the property described in the problem statement.

It is guaranteed that the sum of $n \cdot m$ across all test cases does not exceed 300 000, and the sum of q across test cases does not exceed 100 000.

Output

For each test case, output q integers — the answer to the problem.

Example

standard input	standard output
`1`	`5`
`4 4 8`	`1`
`1 0 0 1`	`0`
`0 2 2 0`	`0`
`0 2 2 0`	`0`
`1 0 0 1`	`0`
`water 0`	`2`
`water -1`	`4`
`water -1`	
`set 2 2 -1`	
`set 3 3 -1`	
`set 3 4 -1`	
`water 1`	
`water 1`	

Note

In the example, after the first height change, the heights remained the same. The heights and islands are shown in the figure below.

1	0	0	1
0	2	2	0
0	2	2	0
1	0	0	1

After the second change, the heights of all cells decrease by 1. Four islands after this drop to height 0, leaving only one island.

0	-1	-1	0
-1	1	1	-1
-1	1	1	-1
0	-1	-1	0

After the third change, the heights decrease by another 1, resulting in no islands remaining.

-1	-2	-2	-1
-2	0	0	-2
-2	0	0	-2
-1	-2	-2	-1

After the fourth, fifth, and sixth changes, there are still no islands, and the heights are as follows:

-1	-2	-2	-1
-2	-1	0	-2
-2	0	-1	-1
-1	-2	-2	-1

After the seventh change, the heights of all cells increase by 1, resulting in the following final heights and islands:

0	-1	-1	0
-1	0	1	-1
-1	1	0	0
0	-1	-1	0

After the eighth change, the heights of all cells increased by 1, resulting in the following final heights and islands:

1	0	0	1
0	1	2	0
0	2	1	1
1	0	0	1

Problem C. Crossword

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

You are given four different words. You need to create a crossword with two horizontal and two vertical words such that the following conditions are met:

- **Each** horizontal word intersects with **each** vertical word;
- Horizontal words are placed in different rows;
- Vertical words are placed in different columns;
- All words must be used exactly **once**;
- Horizontal words in the crossword are read from left to right, vertical words — from top to bottom.

For better understanding of the task, see the examples.

Input

Four different words s_1, s_2, s_3, s_4 are given in four lines ($2 \le |s_i| \le 10$). Each word consists of lowercase letters of the Latin alphabet.

Output

In the first line, output "`Yes`", if it is possible to create a crossword that meets the requirements from the statement, or "`No`" otherwise.

If it is possible to create a crossword, output it as a grid of size 18×18. Empty cells should contain the character "`.`" (dot), and non-empty cells should contain lowercase Latin letters.

If there are multiple possible crosswords, output any of them.

Examples

standard input	standard output
bb aa bba baa	Yes bb................ ba................ aa................ ....
acab babac abbb abbaba	Yes ....b......... ....a.a....... ......abbaba...... ....a.b....... ....acab....... ....
abb bbb baa cbc	No

Problem D. Drinking Coffee is Hard

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

n athletes, numbered with integers from 1 to n, were invited to the finals of the coffee drinking championship. Each athlete is characterized by their *endurance*, which can be expressed as an integer a_i.

During the competition, m cups of coffee, numbered with integers from 1 to m, were placed in a row in front of the athletes. Each cup of coffee is characterized by its *strength*, which can be expressed as an integer b_i. For the convenience of conducting the championship, the organizers arranged the cups in non-decreasing order of the drink's strength; in other words, $b_1 \le b_2 \le \ldots \le b_m$.

It is known that the i-th athlete can drink the j-th cup of coffee if and only if the athlete's endurance is not less than the strength of the drink; that is, $a_i \ge b_j$. Otherwise, the athlete will feel unwell, and the championship will have to be prematurely concluded.

We call a subsegment of cups from the l-th to the r-th ($l \le r$) *feasible* if it is possible to select $r - l + 1$ athletes in such a way that each of them drinks **exactly one** cup of coffee from the set of cups $b_l, b_{l+1}, \ldots, b_r$. At the same time, each cup in the subsegment must be drunk by **exactly one** athlete.

Your task is to find the number of feasible subsegments of cups of coffee.

Input

Each test consists of several test cases. The first line contains one integer t ($1 \le t \le 500\,000$) — the number of test cases. The description of the test cases follows.

The first line of the test case contains two integers n and m ($1 \le n, m \le 500\,000$) — the number of athletes and the number of cups of coffee, respectively.

The second line of the test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the endurance of the athletes.

The third line of the test case contains m integers $b_1, b_2, \ldots, b_m$ ($1 \le b_i \le 10^9$, $b_i \le b_{i+1}$) — the strength of the cups of coffee.

It is guaranteed that the sum of $n + m$ across all test cases does not exceed 10^6.

Output

For each test case, output one integer — the number of feasible subsegments of cups of coffee.

Example

standard input	standard output

3
3 4
1 2 4
1 2 3 4
4 3
2 1 1 4
1 3 4
3 3
1 2 1
1 1 1
	7
	4
	6

Note

In the first test case, the following subsegments are suitable:

1. $l = 1$, $r = 1$: any athlete can drink the coffee;
2. $l = 1$, $r = 2$: the coffee with strengths 1 and 2 will be drunk by athletes with endurance 1 and 2, respectively;
3. $l = 1$, $r = 3$: the coffee with strengths 1, 2, and 3 will be drunk by athletes with endurance 1, 2, and 4, respectively;
4. $l = 2$, $r = 2$: the coffee with strength 2 will be drunk by an athlete with endurance 2;
5. $l = 2$, $r = 3$: the coffee with strengths 2 and 3 will be drunk by athletes with endurance 2 and 4, respectively;
6. $l = 3$, $r = 3$: the coffee with strength 3 will be drunk by an athlete with endurance 4.
7. $l = 4$, $r = 4$: the coffee with strength 4 will be drunk by an athlete with endurance 4.

The subsegment with $l = 1$ and $r = 4$ is not suitable because there will be more cups than athletes.

The subsegment with $l = 2$ and $r = 4$ is not suitable because the coffee with strengths 3 and 4 can only be drunk by one athlete. For a similar reason, the segment with $l = 3$ and $r = 4$ is also not suitable.

Problem E. Enduring the Pokémon

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

The tournament involves n Pokémon, each of which has two characteristics: strength a_i and cost b_i. Before the tournament begins, you can purchase any Pokémon. Then you choose the order in which all other Pokémon will fight your Pokémon one-on-one.

Your Pokémon wins the battle if its strength is strictly greater than that of its opponent; otherwise, your Pokémon loses and leaves the tournament. After each won battle, your Pokémon's strength increases by 1. If your Pokémon defeats all opponents, it wins the tournament.

Find the minimum cost of a Pokémon capable of winning the tournament, or determine that such a Pokémon does not exist.

Input

The first line contains one integer n ($2 \le n \le 100\,000$) — the number of Pokémon.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the strengths of the Pokémon.

The third line contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 10^9$) — the costs of the Pokémon.

Output

Output one integer — the minimum cost of a Pokémon capable of winning the tournament. If such a Pokémon does not exist, output the number -1.

Examples

standard input	standard output

2
1 2
2 1
	1

4
1 1 4 3
1 2 5 4
	4

2
1 1
5 10
	-1

Note

Consider the first example. The Pokémon with number 2 has a strength of 2 and a cost of 1. It is stronger than the first Pokémon with a strength of 1, so it can win the tournament.

Consider the second example. The Pokémon with numbers 1 and 2 have a strength of 1, the lowest among all Pokémon, so they cannot win any battle. The Pokémon with number 4 can win the tournament; it is sufficient to first fight against the Pokémon with number 1, then against the Pokémon with number 2, and finally against the Pokémon with number 3.

The Pokémon with number 3 can also win the tournament, but it is more expensive than the Pokémon with number 4.

In the third example, both Pokémon have the same strength, so neither can win the only possible battle.

Problem F. Fox on the Tree

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

The fox Yae has climbed onto the *tree* of the Sacred Sakura. A tree is defined as a connected undirected graph that contains no cycles.

To move through the tree, the fox uses her magical abilities. In one jump, Yae can leap from vertex v to vertex u if the vertices v and u are connected by an edge, or if there exists a vertex w such that vertices v and w are connected by an edge, and vertices u and w are also connected by an edge.

Currently, Yae is sitting at vertex s of the tree. She noticed an unusual sakura petal at vertex t and now wants to obtain it.

We define a *route* as a sequence of vertices of the tree $v_0, v_1, \ldots, v_k$, such that:

- $v_0 = s$,
- $v_k = t$,
- the fox can make a jump between vertices v_i and v_{i+1},
- all v_i are pairwise distinct.

Help the fox count how many different routes exist in the tree of the Sacred Sakura.

Two routes $v_0, v_1, \ldots, v_k$ and $u_0, u_1, \ldots, u_l$ are considered different if $k \neq l$ or there exists an index i such that $v_i \neq u_i$.

Input

The first line contains three integers n, s, and t ($2 \leq n \leq 200\,000$, $1 \leq s, t \leq n, s \neq t$) — the number of vertices in the tree, the starting vertex, and the ending vertex of the fox's route.

Each of the following $n - 1$ lines contains two integers u_i and v_i ($1 \leq u_i, v_i \leq n$) — the vertices connected by an edge. It is guaranteed that these edges define a tree.

Output

Output a single integer — the number of different routes in the tree. Since the answer can be quite large, output the remainder of the answer when divided by $998\,244\,353$.

Examples

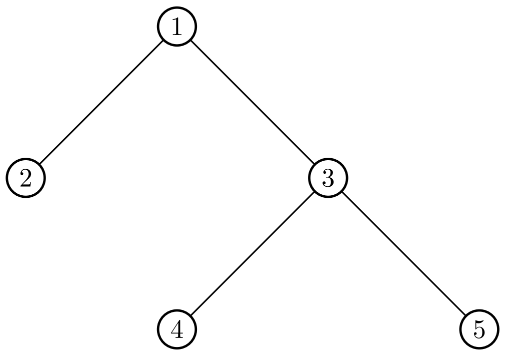
standard input	standard output

`5 1 3`
`1 2`
`1 3`
`3 4`
`3 5`
	`6`

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

Note

Consider the first example. The tree looks as follows:



Possible routes for the fox:

- $1 \to 2 \to 3$
- $1 \to 3$
- $1 \to 4 \to 3$
- $1 \to 4 \to 5 \to 3$
- $1 \to 5 \to 3$
- $1 \to 5 \to 4 \to 3$

Problem G. Good Colorings 6

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

This is an interactive problem.

Little S. bought a carpet, which is represented as a table of size $n \times m$, where each cell is painted in one of three colors — 1, 2, or 3. It is known that any two adjacent cells (sharing a side) are painted **in different colors**. We number the rows of the table from 1 to n from top to bottom, and the columns from 1 to m from left to right. Thus, the cell at the intersection of the i-th row and the j-th column has coordinates (i, j).

Little S. wants to find a *corner*, whose cells are painted in three different colors. A corner consists of three cells with coordinates (i_1, j_1), (i_2, j_2), and (i_3, j_3), such that $i_1 = i_2$, $j_1 = j_3$, $|j_1 - j_2| = 1$, and $|i_1 - i_3| = 1$. In other words, a corner can be represented as a 2×2 square with one of the cells cut out.

Unfortunately, the carpet is very large, and Little S. has poor eyesight, so in one move, Little S. can only discern the color of one arbitrary cell. Your task is to help Little S. find a corner that meets the conditions in no more than 40 moves.

Fortunately, Little S. has already determined that the cell with coordinates (x_1, y_1) is painted in color 1, the cell with coordinates (x_2, y_2) is painted in color 2, and the cell with coordinates (x_3, y_3) is painted in color 3.

Input

Each test consists of several test cases. The first line contains a single integer t ($1 \le t \le 1\,000$) — the number of test cases.

Interaction Protocol

Each test case begins with your program reading eight integers n, m, x_1, y_1, x_2, y_2, x_3, and y_3 ($2 \le n, m \le 10^9$, $1 \le x_i \le n$, $1 \le y_i \le m$) — the dimensions of the carpet, the coordinates of the cell of color 1, the coordinates of the cell of color 2, and the coordinates of the cell of color 3.

After this, your program can make queries of two types:

- `ask` x y ($1 \le x \le n$, $1 \le y \le m$) — a query for the color of the cell. In response to this query, the jury's program outputs a single number from 1 to 3 — the color of the cell with coordinates (x, y). For each test case, you can make no more than 40 cell color queries.
- `done` — the end of interaction for the current test case. At the moment this query is executed, there must exist a corner of three cells of different colors among the cells with **known** colors (including the cells (x_1, y_1), (x_2, y_2), and (x_3, y_3)). After this query, the interaction for the current test case ends, and interaction for the next test case (if any) begins.

If you use "`cout « ... « endl`" in C++, "`System.out.println`" in Java, "`print`" in Python, "`writeln`" in Pascal, the output stream is flushed automatically, and you do not need to do anything extra.

If you use another method of output, it is recommended to flush the output stream buffer. Note that a newline must be output in any case. To flush the output stream buffer, you can use "`fflush(stdout)`" in C++, "`System.out.flush()`" in Java, "`sys.stdout.flush()`" in Python.

The interactor is not adaptive. In other words, the colors of all cells are fixed in advance.

Example

standard input	standard output
1	
4 3 1 1 4 1 1 3	
	ask 1 2
2	
	ask 2 1
2	
	ask 2 2
1	
	done

Note

In the example, the colors of the following cells are initially known:

1	?	3
?	?	?
?	?	?
2	?	?

After three queries, the colors of the following cells are known:

1	2	3
2	1	?
?	?	?
2	?	?

A corner with cells at coordinates: $(1, 2)$, $(1, 3)$, and $(2, 2)$ has cells of three different colors, and therefore is suitable.

Problem H. Hashing

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

We define the *hash* with base k and modulus p of a sequence of numbers $s_1, s_2, \ldots, s_m$ as follows: $(s_1 \cdot k^{m-1} + s_2 \cdot k^{m-2} + \ldots + s_{m-1} \cdot k + s_m \cdot 1) \bmod p$.

You are given an array of n integers $a_1, a_2, \ldots, a_n$. Find the number of subarrays of the array whose hash equals x. In other words, you need to find the number of pairs $1 \le l \le r \le n$ such that $(a_l \cdot k^{r-l} + a_{l+1} \cdot k^{r-l-1} + \ldots + a_r) \bmod p = x$.

Input

The first line contains four integers n, k, p, and x ($1 \le n \le 200\,000$, $1 \le k < p \le 10^9$, $0 \le x < p$) — the length of the array, the base of the hash, the modulus of the hash, and the required value. It is guaranteed that p is a prime number.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i < k$) — the elements of the array.

Output

Output a single integer — the number of subarrays of the array whose hash equals x.

Examples

standard input	standard output

`4 3 7 3`
`1 0 1 2`
	`2`

`5 2 3 1`
`1 0 0 1 0`
	`5`

Note

In the first example, two subarrays are suitable: $[1, 3]$ and $[1, 2]$.

In the second example, five subarrays are suitable: $[1, 1]$, $[1, 3]$, $[2, 4]$, $[3, 4]$, and $[4, 4]$.

Problem I. Industrial Robots

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

Two robots, L and R, find themselves in a cave represented by a sequence of n columns of width 1. In the i-th column, the ceiling of the cave has a height of h_i. At the beginning of the first second, robot L is in the first column, while robot R is in the n-th.

A collapse begins in the cave. Every second, the following occurs:

- The ceiling in the cave drops by 1. In other words, all positive h_i decrease by 1. If the height of the ceiling in the column where a robot is standing becomes 0, then the robot gets crushed by the ceiling and breaks. If one of the robots is broken or the robots cannot see each other (i.e., there is a column with a zero ceiling height between the columns where they are standing), both robots shut down and the process ends. Note that if the robots are in the same column, they can see each other;
- If neither robot is broken, they can see each other, and they are not standing in the same column, then exactly one of the robots moves to an adjacent column to get closer to the other robot. In other words, either robot L moves one column to the right, or robot R moves one column to the left. If the robots are in the same column, nothing happens.

Count the number of pairs (l, r) such that $1 \le l \le r \le n$ and there exists some way to move the robots every second so that at the beginning of some second, robot L is in column l, robot R is in column r, neither robot is broken, and the robots can see each other.

Input

Each test consists of several test cases. The first line contains a single integer t ($1 \le t \le 10\,000$) — the number of test cases. The description of the test cases follows.

The first line of a test case contains a single integer n ($1 \le n \le 500\,000$) — the number of columns in the cave.

The second line of a test case contains n integers $h_1, h_2, \ldots, h_n$ ($1 \le h_i \le n$) — the heights of the ceilings in the columns of the cave.

It is guaranteed that the sum of n across all test cases does not exceed $500\,000$.

Output

For each test case, output a single integer — the number of suitable pairs (l, r).

Example

standard input	standard output

2
5
2 4 4 3 5
10
10 10 9 10 10 7 10 10 10 10
	5
	48

Note

Consider the first test case. It consists of 5 columns with initial heights of $[2, 4, 4, 3, 5]$.

At the beginning of the first second, robots L and R are in the first and fifth columns, respectively, so the pair $(1, 5)$ is counted in the answer.

Next, the height of the ceiling in the cave decreases by 1, and the heights of the columns become $[1, 3, 3, 2, 4]$. All heights are positive, so the robots can see each other. After this, either robot L moves to the second column, or robot R moves to the fourth column. Thus, two cases are possible:

- Robot L moves to the second column, and at the beginning of the second second, the robots are in columns 2 and 5. They can see each other, so the pair $(2, 5)$ is counted in the answer.

  Next, the height of the ceiling decreases by 1, so the heights of the cave columns are $[0, 2, 2, 1, 3]$. The robots can see each other, so the process continues, and one of the robots must move one column closer to the other.

  If robot L moves to the third column, then the robots will be in columns 3 and 5 at the beginning of the third second; otherwise, they will be in columns 2 and 4. Thus, the pairs $(3, 5)$ and $(2, 4)$ are counted in the answer.

  At the third second, the heights of the ceilings will decrease by 1, resulting in $[0, 1, 1, 0, 2]$. If the robots are in columns 3 and 5, they will not be able to see each other due to the zero height of the ceiling in the fourth column, and if they are in columns 2 and 4, robot R will be crushed by the ceiling and broken.

  Thus, regardless of the robots' movements, the process will stop at the third second;

- Robot R moves to the fourth column, and at the beginning of the second second, the robots are in columns 1 and 4. They can see each other, so the pair $(1, 4)$ is counted in the answer. Next, the height of the ceiling in the first column will become 0, so robot L will be broken and the process will stop.

Thus, we find that the counted pairs are $(1, 4)$, $(1, 5)$, $(2, 4)$, $(2, 5)$, and $(3, 5)$, so the answer is 5.

Problem J. Just a Turtle Problem

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

You are given an array a, consisting of n integers $a_1, a_2, \ldots, a_n$. We will construct a table of size $n \times n$ according to the following rules:

- The first row of the table is equal to $a_1, a_2, \ldots, a_n$;
- The i-th row of the table is equal to the *left cyclic shift* of the $(i-1)$-th row of the table.

A left cyclic shift of the array $b_1, b_2, \ldots, b_{n-1}, b_n$ is the array $b_2, \ldots, b_n, b_1$.

In cell $(1, 1)$ of the resulting table, there is a turtle. In one move, it can either move one cell to the right or one cell down. In other words, if the turtle is in cell (i, j), it can move either to cell $(i, j+1)$ (if it exists) or to cell $(i+1, j)$ (if it exists). The goal of the turtle is to reach cell (n, n).

Your task is to help the turtle find a path from cell $(1, 1)$ to cell (n, n) with the **minimum** sum of the numbers written in the cells it visits.

Input

The first line contains one integer n ($1 \le n \le 100\,000$) — the size of the array a.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($|a_i| \le 10^9$) — the elements of the array a.

Output

Output one integer — the minimum sum of the numbers written in the cells visited by the turtle.

Examples

standard input	standard output
1 10	10
2 1 -1	1

Note

In the first example, the table consists of one element, equal to 10. Thus, the turtle does not need to make any moves to reach the goal.

In the second example, the first row of the table is equal to $[1, -1]$, and the second row of the table is equal to $[-1, 1]$. The turtle has two possible paths: $(1, 1) \to (1, 2) \to (2, 2)$ and $(1, 1) \to (2, 1) \to (2, 2)$. The sum of the numbers in the visited cells is $1 + (-1) + 1 = 1$ in both cases.

Problem K. Keeping the Medians Close 1

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Two arrays a and b of **odd** length n are given. All numbers in the arrays are pairwise distinct.

Let median(c) denote the median of array c, which is the element that, after sorting array c in ascending order, will be at position $\frac{n+1}{2}$ when numbering elements starting from one.

The task is to minimize the value of $|\text{median}(a) - \text{median}(b)|$ after performing any number of operations of the following type:

- Choose i from 1 to n inclusive and swap a_i and b_i.

Input

Each test consists of several test cases. The first line contains a single integer t ($1 \le t \le 10\,000$) — the number of test cases. The description of the test cases follows.

The first line of the test case contains a single integer n ($1 \le n < 200\,000$) — the length of the arrays. It is guaranteed that n is odd.

The second line of the test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the elements of array a.

The third line of the test case contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 10^9$) — the elements of array b.

It is guaranteed that all elements in arrays a and b are pairwise distinct. That is, $a_i \neq b_j$ for any i and j, as well as $a_i \neq a_j$ (for $i \neq j$) and $b_i \neq b_j$ (for $i \neq j$).

It is guaranteed that the sum of n across all test cases does not exceed $200\,000$.

Output

For each test case, output two lines containing the arrays that minimize the value of $|\text{median}(a) - \text{median}(b)|$.

In the first line, output n integers a_1, a_2, …, a_n — the elements of array a after performing the operations.

In the second line, output n integers b_1, b_2, …, b_n — the elements of array b after performing the operations.

Example

standard input	standard output

```
3
3
1 2 3
4 5 6
5
20 16 19 7 4
1 15 2 8 10
7
5 10 20 13 29 1 14
22 28 16 27 23 15 21
```
	```
	1 5 3
	4 2 6
	20 15 2 8 4
	1 16 19 7 10
	22 28 16 13 23 1 14
	5 10 20 27 29 15 21
	```

Note

In the first test case, after swapping a_2 and b_2, the answer is $|3 - 4| = 1$ and is optimal.

Problem L. Keeping the Medians Close 2

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Two arrays a and b of **odd** length n are given. All numbers in the arrays are pairwise distinct. An integer k from 0 to n is also given.

We denote median(c) as the median of the array c, which is the element that, after sorting the array c in ascending order, will be at position $\frac{n+1}{2}$ when numbering elements from one.

The task is to find the minimum value of $|\text{median}(a) - \text{median}(b)|$ after performing no more than k operations of the following type:

- Choose i from 1 to n inclusive and swap a_i and b_i.

Input

Each test consists of several test cases. The first line contains one integer t ($1 \le t \le 10\,000$) — the number of test cases. The description of the test cases follows.

The first line of the test case contains two integers n and k ($1 \le n < 200\,000$, $0 \le k \le n$) — the length of the arrays and the limit on the number of operations. It is guaranteed that n is odd.

The second line of the test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the elements of array a.

The third line of the test case contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 10^9$) — the elements of array b.

It is guaranteed that all elements in arrays a and b are pairwise distinct. That is, $a_i \ne b_j$ for any i and j, and also $a_i \ne a_j$ (for $i \ne j$) and $b_i \ne b_j$ (for $i \ne j$).

It is guaranteed that the sum of n across all test cases does not exceed $200\,000$.

Output

For each test case, output one integer — the minimum value of $|\text{median}(a) - \text{median}(b)|$ after performing no more than k operations.

Example

standard input	standard output

4
3 0
1 2 3
4 5 6
3 1
1 2 3
4 5 6
5 1
20 16 19 7 4
1 15 2 8 10
7 2
5 10 20 13 29 1 14
22 28 16 27 23 15 21
	3
	1
	3
	4

Note

In the first test case, $k = 0$, so no operations can be performed. The median of the first array is 2, and the median of the second array is 5, thus the answer is $|2 - 5| = 3$.

In the second test case, $k = 1$, so one swap can be made. If we swap a_2 and b_2, then the median of the first array will be 3, and the median of the second array will be 4, thus the answer will be 1.

Problem M. Magnification Matrix

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

At the Skolkovo Institute of Nanotechnology, researchers are experimenting with a particle amplification grid.

This grid is composed of nanoparticles arranged in a rectangular grid. Each functional nanoparticle receives charged particles from the left or bottom and emits one particle upward and one to the right. These particles then continue to propagate if the destination node exists and is not faulty.

However, due to imperfections in the manufacturing process, some nanoparticles are defective: they neither receive nor emit particles.

Some of the faulty positions are known in advance. Every other node in the grid works correctly with probability p, and is faulty with probability $1 - p$.

The manufacturing process allows control over p by adjusting conditions such as chamber pressure.

Your task is to find the value of p such that a single particle injected into the bottom-left corner of the grid will, in expectation, result in exactly k particles reaching the top-right corner of the grid — or determine that this is impossible.

Note: if the node in the top-right corner is faulty, it cannot receive any particles.

Input

The first line contains four integers w, h, n, and k ($1 \le w, h \le 5\,000$, $0 \le n \le 50$, $1 \le k \le 10^{10000}$), representing the width and height of the grid, the number of confirmed faulty nodes, and the desired expected number of particles received at node $(w-1, h-1)$.

The next n lines each contain two integers x and y ($0 \le x < w$, $0 \le y < h$), indicating the coordinates of a known faulty node. All positions are distinct.

Output

Print one real number — the required probability p.

Your answer will be accepted if the absolute or relative error does not exceed 10^{-6}.

If it is impossible to reach the expected number of particles for any p in $[0, 1]$, print `-1`.

Examples

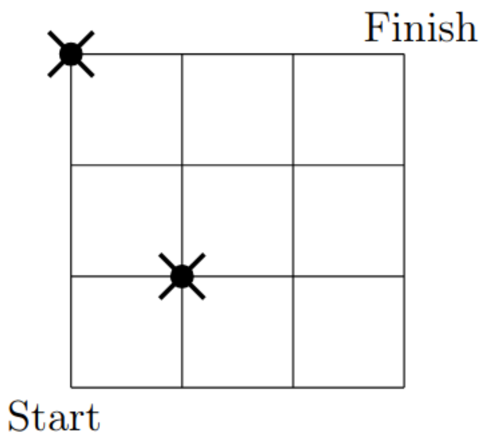
standard input	standard output

`4 4 2 5`
`0 3`
`1 1`
	`0.953069488577940520128`

`3 4 1 10`
`0 1`
	`-1`

Note

In the first example, a particle starts at the bottom-left corner ("Start") and, depending on the configuration and the probability p, enough amplified particles can reach the top-right corner ("Finish").



In the second example, even under perfect conditions ($p = 1$), only 4 particles can reach the final node — making it impossible to achieve 10.