The 2024 Kunming Invitational Contest

Contest Session

May 26, 2024

Problem List

A	Two-star Contest
B	Gold Medal
C	Stop the Castle 2
D	Generated String
E	Relearn through Review
F	Collect the Coins
G	Be Positive
H	Subarray
I	Left Shifting 2
J	The Quest for El Dorado
K	Permutation
L	Trails
M	Italian Cuisine

This problem set should contain 13 (thirteen) problems on 17 (seventeen) numbered pages. Please inform a runner immediately if something is missing from your problem set.

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

Hosted by

Problem Set Prepared by

It's against the rules to open non-contest websites during the contest.
If you're interested (which is our pleasure),
please scan the QR code only after the contest.

Problem A. Two-star Contest

Education experts are going to rate n contests due to whatever reasons. The experts have already decided on the rating of each contest, where the i-th contest is rated as a s_i-star contest.

It is said that each contest is rated according to m properties, where the j-th property of the i-th contest is indicated as $p_{i,j}$, whose value ranges from 0 to k (both inclusive). The score of a contest is calculated as the sum of all its m properties. That is, let v_i be the score of the i-th contest, we have $v_i = \sum_{j=1}^{m} p_{i,j}$.

It would be natural for a contest with more stars to have a higher score. The experts require that, for any two contests $1 \le i, j \le n$, if $s_i > s_j$, then there must be $v_i > v_j$. Unfortunately, the experts forgot to collect values for some (or even all) properties of some contests. As the assistant of the experts, you're required to fill in the missing values so that the constraint stated above holds for any two contests.

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 three integers n, m, and k ($2 \le n \le 4 \times 10^5$, $1 \le m \le 4 \times 10^5$, $n \times m \le 4 \times 10^5$, $1 \le k \le 10^9$) indicating the number of contests, the number of properties of each contest, and the upper bound of the value of each property.

For the following n lines, the i-th line first contains an integer s_i ($1 \le s_i \le 10^9$) indicating the number of stars the i-th contest is rated. Then m integers $p_{i,1}, p_{i,2}, \cdots, p_{i,m}$ follow ($-1 \le p_{i,j} \le k$). If $p_{i,j} = -1$ then the value of the j-th property of the i-th contest is missing and it is your quest to fill it in; Otherwise if $p_{i,j} \ge 0$ then the value of the j-th property of the i-th contest is given and you should not change it.

It's guaranteed that the sum of $n \times m$ of all test cases will not exceed 4×10^5.

Output

For each test case:

If it is possible to fill in all the missing properties while satisfying the constraint, first output `Yes` in one line. Then output n lines where the i-th line contains m integers $q_{i,1}, q_{i,2}, \cdots, q_{i,m}$ ($0 \le q_{i,j} \le k$) separated by a space, indicating the m properties of the i-th contest after you fill in the missing values. If $p_{i,j} = -1$ then $q_{i,j}$ should be the value you fill in; Otherwise if $p_{i,j} \ge 0$ then $q_{i,j} = p_{i,j}$. If there are multiple valid answers, you can output any of them.

If it is not possible to satisfy the constraint, just output `No` in one line.

Example

standard input	standard output
5 3 4 5 5 1 3 -1 -1 2 -1 5 -1 5 3 3 -1 -1 4 2 3 10 10000 5 0 -1 1 10 10 10 2 3 10 10 1 2 3 100 4 5 6 2 3 10 100 1 2 3 10 4 5 6 2 3 10000 100 -1 -1 -1 1 -1 -1 -1	Yes 1 3 5 4 0 5 0 5 3 3 2 4 No Yes 1 2 3 4 5 6 No Yes 2024 5 26 11 45 14

Note

For the second sample test case, even if we fill the only -1 with the maximum possible value 10, the score of the first contest is still only 15, which is not larger than the score 30 of the second contest.

Problem B. Gold Medal

What a busy week! This weekend, n programming contests will be held at the same time.

Each contest will award one gold medal for every k participating teams. That is to say, if there are t teams participating in the contest, $\lfloor \frac{t}{k} \rfloor$ gold medals will be awarded, where $\lfloor x \rfloor$ is the largest integer not exceeding x. Currently, there will be a_i teams participating in the i-th contest.

BaoBao is the coach of a university with m teams, and he hasn't decided which contests these teams should participate in. Please help him assign each team to a contest, so that the total number of gold medals handed out by all contests is maximized.

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 line contains two integers n and k ($1 \le n \le 100$, $1 \le k \le 10^9$), indicating the number of contests and the ratio of gold medals.

The second line contains n integers $a_1, a_2, \cdots, a_n$ ($1 \le a_i \le 10^9$), where a_i is the current number of teams participating in the i-th contest.

The third line contains an integer m ($1 \le m \le 10^9$), indicating the number of teams you need to assign a contest to.

Output

For each test case output one line containing one integer indicating the maximum total number of gold medals handed out by all contests.

Example

standard input	standard output
2 3 10 239 141 526 6 2 1 300 100 1000	91 1400

Note

For the first sample test case, send 2 teams to the 1-st contest and 4 teams to the 3-rd contest. The total number of gold medals will be $\lfloor \frac{239+2}{10} \rfloor + \lfloor \frac{141+0}{10} \rfloor + \lfloor \frac{526+4}{10} \rfloor = 24 + 14 + 53 = 91$.

Problem C. Stop the Castle 2

There are n castles and m obstacles on a chessboard with 10^9 rows and 10^9 columns. Each castle or obstacle occupies exactly one cell and all occupied cells are distinct. Two castles can attack each other, if they're on the same row or the same column, and there are no obstacles or other castles between them. More formally, let (i, j) be the cell on the i-th row and the j-th column. Two castles positioned at (i_1, j_1) and (i_2, j_2) can attack each other, if one of the following conditions is true:

- $i_1 = i_2$ and for all $\min(j_1, j_2) < j < \max(j_1, j_2)$, there is no obstacle or castle positioned at (i_1, j).
- $j_1 = j_2$ and for all $\min(i_1, i_2) < i < \max(i_1, i_2)$, there is no obstacle or castle positioned at (i, j_1).

You have to remove k obstacles from the chessboard, but you don't want too many castles to attack each other. Minimize the number of pairs of castles which can attack each other after removing exactly k obstacles from the chessboard.

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 three integers n, m, and k ($1 \le n, m \le 10^5$, $1 \le k \le m$) indicating the number of castles, the number of obstacles, and the number of obstacles you have to remove.

For the following n lines, the i-th line contains two integers r_i and c_i ($1 \le r_i, c_i \le 10^9$) indicating that the i-th castle is located on the r_i-th row and the c_i-th column.

For the following m lines, the i-th line contains two integers r'_i and c'_i ($1 \le r'_i, c'_i \le 10^9$) indicating that the i-th obstacle is located on the r'_i-th row and the c'_i-th column.

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

Output

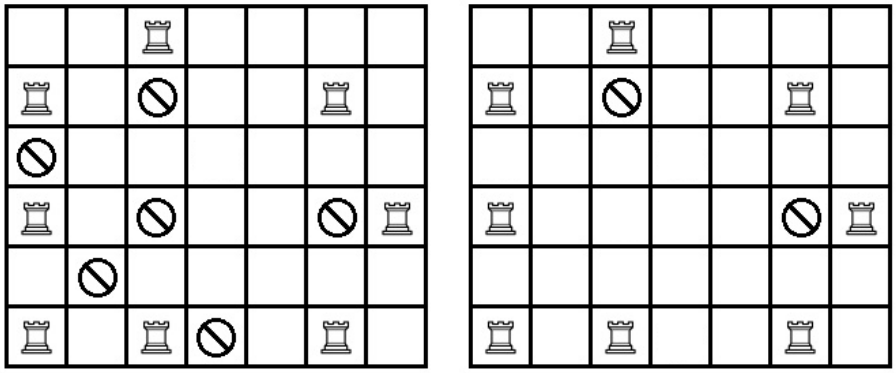
For each test case, first output one line containing one integer indicating the smallest number of pairs of castles which can attack each other after removing exactly k obstacles. Then output another line containing k distinct integers $b_1, b_2, \cdots, b_k$ ($1 \le b_i \le m$) separated by a space, indicating the indices of obstacles you want to remove. If there are multiple valid answers, you can output any of them.

Example

standard input	standard output
3 8 6 4 1 3 2 1 2 6 4 1 4 7 6 1 6 3 6 6 2 3 3 1 4 3 4 6 5 2 6 4 3 2 1 10 12 10 10 10 11 1 4 1 5 1 3 2 1 1 2 1 2 2 2 3	4 6 3 2 5 2 1 0 1 2

Note

For the first sample test case, the image on the left shows the original chessboard, and the image on the right shows the chessboard after removing 4 obstacles. After removing the obstacles, the pairs of castles which can attack each other are: the 2-nd and the 4-th castles, the 4-th and the 6-th castles, the 6-th and the 7-th castles, the 7-th and the 8-th castles.



For the third sample test case, as there is only 1 castle, there is no pair of castles which can attack each other.

Problem D. Generated String

You are given a template string $S = s_1 s_2 \cdots s_n$ of length n. A generated string is a string formed by concatenating several substrings of S. Formally, each generated string $T = f(k, \{l_i\}_{i=1}^k, \{r_i\}_{i=1}^k)$ is described by a positive integer k and k pairs of integers (l_i, r_i), where $T = s[l_1 : r_1] + s[l_2 : r_2] + \cdots + s[l_k : r_k]$. Here $s[l : r]$ denotes the substring $s_l s_{l+1} \cdots s_r$, and $+$ denotes string concatenation.

Your task is to maintain a multiset $\mathbb{A}$ of strings, supporting the following three types of operations:

- `+` $k\ l_1\ r_1\ l_2\ r_2\ \cdots\ l_k\ r_k$: Insert $f(k, \{l_i\}_{i=1}^k, \{r_i\}_{i=1}^k)$ into the multiset $\mathbb{A}$.
- `-` t: Erase the string inserted in the t-th operation from the multiset $\mathbb{A}$. It is guaranteed that the t-th operation is an inserting operation and the inserted string is not erased at this time.
- `?` $k\ l_1\ r_1\ l_2\ r_2\ \cdots\ l_k\ r_k\ m\ u_1\ v_1\ u_2\ v_2\ \cdots u_m\ v_m$: Answer the number of strings in the multiset $\mathbb{A}$ that begin with string $f(k, \{l_i\}_{i=1}^k, \{r_i\}_{i=1}^k)$ and end with $f(m, \{u_i\}_{i=1}^m, \{v_i\}_{i=1}^m)$.

Input

There is only one test case in each test file.

The first line contains two integers n and q ($1 \le n, q \le 10^5$), indicating the length of S and the number of operations.

The second line contains a string $s_1 s_2 \cdots s_n$ consisting of lower-cased English letters, indicating the template string.

For the following q lines, the i-th line contains an operation in the format described above. It is guaranteed that $1 \le l_i \le r_i \le n$, $1 \le u_i \le v_i \le n$. It's also guaranteed that the sum of k in all operations of type `+`, plus the sum of k in all operations of type `?`, plus the sum of m in all operations of type `?`, will not exceed 3×10^5.

Output

For each operation of type `?`, output one line containing one integer indicating the answer.

Example

standard input	standard output
8 7 abcaabbc + 3 1 3 2 4 3 8 + 2 1 4 1 8 + 1 2 4 ? 1 5 6 1 7 8 - 3 + 1 2 5 ? 1 2 3 1 5 5	2 1

Problem E. Relearn through Review

Given an integer sequence $a_1, a_2, \cdots, a_n$ of length n and a non-negative integer k, you can perform the following operation at most once: Choose two integers l and r such that $1 \le l \le r \le n$, then for each $l \le i \le r$, change a_i into $(a_i + k)$.

Maximize the greatest common divisor of the whole sequence.

An integer g is said to be the common divisor of the whole sequence, if a_i is divisible by g for all $1 \le i \le n$.

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 ($1 \le n \le 3 \times 10^5$, $0 \le k \le 10^{18}$).

The second line contains n integers $a_1, a_2, \cdots, a_n$ ($1 \le a_i \le 10^{18}$) indicating the sequence.

It's guaranteed that the sum of n of all test cases does not exceed 3×10^5.

Output

For each test case output one line containing one integer indicating the maximum greatest common divisor of the whole sequence.

Example

standard input	standard output
2 6 2 5 3 13 8 10 555 3 0 3 6 9	5 3

Note

For the first sample test case, choose $l = 2$ and $r = 4$. The sequence will become $\{5, 5, 15, 10, 10, 555\}$. Its greatest common divisor is 5.

Problem F. Collect the Coins

There are 10^9 cells arranged in a row, numbered from 1 to 10^9 from left to right. Two robots are patrolling in the cells. The maximum speed of each robot is v cells per second (v is an integer), indicating that if a robot is currently in cell p, it can move to any cell p' in the next second, as long as $1 \le p' \le n$ and $|p' - p| \le v$.

There will be n coins appearing in the cells. The i-th coin will appear at the t_i-th second in cell c_i. If a robot is in the same cell at that time, it can pick up the coin. Otherwise the coin will instantly disappear.

More formally, during each second, the following steps will happen in order:

- Each robot can move to a position no more than v cells away (it is OK to just stay in its current cell).
- Coins appear in the cells.
- If at least one robot is in the same cell with a coin, that coin is collected.
- All uncollected coins disappear.

Your task is to determine the initial positions of two robots before the 1-st second, and move them wisely to collect all the coins with the smallest possible v. Output this optimal value of v.

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 ($1 \le n \le 10^6$) indicating the number of coins appearing in the cells.

For the following n lines, the i-th line contains two integers t_i and c_i ($1 \le t_i, c_i \le 10^9$) indicating the time when the i-th coin appears and the position where the i-th coin appears. It's guaranteed that $t_i \le t_{i+1}$ for all $1 \le i < n$. It's also guaranteed that $t_i \ne t_j$ or $c_i \ne c_j$ for all $i \ne j$.

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

Output

For each test case, output one line containing one integer, indicating the smallest possible maximum speed of the robots. If it's impossible to collect all the coins, output `-1` instead.

Example

standard input	standard output
3 5 1 1 3 7 3 4 4 3 5 10 1 10 100 3 10 100 10 1000 10 10000	2 0 -1

Problem G. Be Positive

Find the lexicographically smallest permutation $p_0, p_1, p_2 \cdots, p_{n-1}$ of $0, 1, 2, \cdots, (n-1)$ satisfying the following constraint: For all $0 \le i < n$, $p_0 \oplus p_1 \oplus \cdots \oplus p_i > 0$. Here $\oplus$ is the bitwise exclusive or operation.

We say an integer sequence $p_0, p_1, \cdots, p_{n-1}$ of length n is lexicographically smaller than another integer sequence $q_0, q_1, \cdots, q_{n-1}$ of length n, if there exists an integer $0 \le k < n$ such that $p_i = q_i$ for all $0 \le i < k$ and $p_k < q_k$.

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 and only line contains an integer n ($1 \le n \le 10^6$).

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

Output

For each test case output one line containing n integers separated by a space, indicating the lexicographically smallest permutation satisfying the constraint. If there is no such permutation, output `impossible` instead.

Example

standard input	standard output
4 1 2 3 4	impossible 1 0 1 0 2 impossible

Problem H. Subarray

Given an integer sequence $a_1, a_2, \cdots, a_n$ of length n, we say a continuous subarray $a_l, a_{l+1}, \cdots, a_r$ is good, if the maximum element of the subarray appears exactly k times in the subarray. For each $1 \le k \le n$, count the number of good subarrays.

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 ($1 \le n \le 4 \times 10^5$) indicating the length of the sequence.

The second line contains n integers $a_1, a_2, \cdots, a_n$ ($1 \le a_i \le 10^9$) indicating the sequence.

It's guaranteed that the sum of n of all test cases does not exceed 4×10^5.

Output

Let c_i be the number of good subarrays when $k = i$. To decrease the size of output, for each test case, just output one line containing one integer indicating $\sum_{i=1}^{n}(i \times c_i^2)$ modulo 998244353.

Example

standard input	standard output
3 11 1 1 2 1 2 2 3 3 2 3 1 3 2024 5 26 3 1000000000 1000000000 1000000000	2564 36 20

Note

We denote $[l, r]$ as the continuous subarray $a_l, a_{l+1}, \cdots, a_r$. For the first sample test case:

- $c_1 = 27$, $c_2 = 22$ and $c_3 = 17$, while $c_4, c_5, \cdots, c_{11}$ are all 0. So the answer is $(1 \times 27^2 + 2 \times 22^2 + 3 \times 17^2) \bmod 998244353 = 2564$.
- Some examples of good subarrays when $k = 1$ are $[3, 3]$ (maximum element 2 appears once), $[4, 5]$ (maximum element 2 appears once), and $[9, 10]$ (maximum element 3 appears once).
- Some examples of good subarrays when $k = 2$ are $[1, 2]$ (maximum element 1 appears twice), $[4, 6]$ (maximum element 2 appears twice), and $[6, 9]$ (maximum element 3 appears twice).
- Some examples of good subarrays when $k = 3$ are $[3, 6]$ (maximum element 2 appears three times), $[2, 6]$ (maximum element 2 appears three times), and $[1, 11]$ (maximum element 3 appears three times).

Problem I. Left Shifting 2

Given a string consisting of lower-cased English letters, we say the string is beautiful if all neighboring characters are different. For example, `icpc` and `kunming` are beautiful but `hello` is not, because its 3-rd and 4-th characters are the same.

Given a string $S = s_0 s_1 \cdots s_{n-1}$ of length n consisting of lower-cased English letters, let $f(S, d)$ be the string obtained by shifting S to the left d times. That is $f(S, d) = s_{(d+0) \bmod n} s_{(d+1) \bmod n} \cdots s_{(d+n-1) \bmod n}$.

Let $g(S, d)$ be the smallest number of operations needed to make string $f(S, d)$ beautiful. In each operation, you can change one character in $f(S, d)$ to any lower-cased English letter.

Find a non-negative integer d which minimizes $g(S, d)$ and output the minimized value.

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 and only line contains a string $s_0 s_1 \cdots s_{n-1}$ ($1 \le n \le 5 \times 10^5$) consisting only of lower-cased English letters.

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

Output

For each test case, output one line containing one integer, indicating the smallest possible $g(S, d)$.

Example

standard input	standard output
3 abccbbbbd abcde x	2 0 0

Note

For the first sample test case, consider $d = 5$. We have $f(S, 5) =$ `bbbdabccb`. For this string, we can change its 2-nd character to `x` and its 8-th character to `y`, so the string will become `bxbdabcyb`, which is a beautiful string.

Problem J. The Quest for El Dorado

There is a kingdom with n cities and m bi-directional railways connecting the cities. The i-th railway is operated by the c_i-th railway company, and the length of the railway is l_i.

You'd like to travel around the country starting from city 1. For your travel, you have bought k railway tickets. The i-th ticket can be represented by two integers a_i and b_i, meaning that if you use this ticket, you can travel through some railways in one go, as long as they're all operated by company a_i and their total length does not exceed b_i. It is also allowed to just stay in the current city when using a ticket. You can only use the tickets one at a time, and you can only use each ticket once.

As you find it a burden to determine the order to use the tickets, you decided to use the tickets just in their current order. More formally, you're going to perform k operations. In the i-th operation, you can either choose to stay in the current city u; Or choose a different city v, such that there exists a path from city u to city v where all railways in the path are operated by company a_i and the total length of railways does not exceed b_i, and finally move to city v.

For each city, determine if it is possible to arrive in that city after using all k tickets.

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 three integers n, m, and k ($2 \le n \le 5 \times 10^5$, $1 \le m \le 5 \times 10^5$, $1 \le k \le 5 \times 10^5$) indicating the number of cities, the number of railways and the number of tickets.

For the following m lines, the i-th line contains four integers u_i, v_i, c_i, and l_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$, $1 \le c_i \le m$, $1 \le l_i \le 10^9$) indicating that the i-th railway connects city u_i and v_i. It is operated by company c_i and its length is l_i. Note that there might be multiple railways connecting the same pair of cities.

For the following k lines, the i-th line contains two integers a_i and b_i ($1 \le a_i \le m$, $1 \le b_i \le 10^9$) indicating that if you use the i-th ticket, you can travel through some railways in one go, if they're all operated by company a_i and their total length does not exceed b_i.

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

Output

For each test case, output one line containing one string $s_1 s_2 \cdots s_n$ of length n where each character is either 0 or 1. If you can travel from city 1 to city i with these k tickets, then $s_i = 1$; Otherwise $s_i = 0$.

Example

standard input	standard output
2 5 6 4 1 2 1 30 2 3 1 50 2 5 5 50 3 4 6 10 2 4 5 30 2 5 1 40 1 70 6 100 5 40 1 30 3 1 1 2 3 1 10 1 100	11011 100

Note

For the first sample test case:

- To arrive in city 4, you can use the 1-st ticket to move from city 1 to city 2, then stay in city 2 when using the 2-nd ticket, then use the 3-rd ticket to move from city 2 to city 4, then stay in city 4 when using the 4-th ticket.
- To arrive in city 5, you can use the 1-st ticket to move from city 1 to city 5 by going through the 1-st and the 6-th railway, then stay in city 5 when using the following tickets.
- As you cannot change the order to use the tickets, you cannot arrive in city 3.

Problem K. Permutation

This is an interactive problem.

There is a hidden permutation of n. Recall that a permutation of n is a sequence where each integer from 1 to n (both inclusive) appears exactly once. Piggy wants to unravel the permutation with some queries.

Each query must consist of a sequence (not necessarily a permutation) with n integers ranging from 1 to n (both inclusive). Each query is answered with an integer x, indicating the number of the positions where the corresponding element in Piggy's query sequence matches that of the hidden permutation. For example, if the hidden permutation is $\{1, 3, 4, 2, 5\}$ and the sequence Piggy asks is $\{2, 3, 5, 2, 5\}$, he'll receive 3 as the answer.

As Piggy is busy recently, he gives this problem to you. Find the permutation with no more than 6666 queries.

Input

There is only one test case in each test file.

The first line of the input contains an integer n ($1 \le n \le 10^3$) indicating the length of the hidden permutation.

Interaction Protocol

To ask a query, output one line. First output `0` followed by a space, then print a sequence of n integers ranging from 1 to n separated by a space. After flushing your output, your program should read a single integer x indicating the answer to your query.

If you want to guess the permutation, output one line. First output `1` followed by a space, then print a permutation of n separated by a space. After flushing your output, your program should exit immediately.

Note that the answer for each test case is pre-determined. That is, the interactor is not adaptive. Also note that your guess does not count as a query.

To flush your output, you can use:

- `fflush(stdout)` (if you use `printf`) or `cout.flush()` (if you use `cout`) in C and C++.
- `System.out.flush()` in Java.
- `stdout.flush()` in Python.

Example

standard input	standard output
5 3 4 2	 0 3 1 3 2 2 0 3 1 5 2 2 0 3 5 4 4 4 1 3 1 5 2 4

Note

Please note that if you receive a Time Limit Exceeded verdict, it is possible that your query is invalid or the number of queries exceeds the limit.

Problem L. Trails

BaoBao is taking a walk on an infinite two-dimensional plane. For each point (x, y) on the plane where both x and y are integers, there is a bi-directional trail connecting points (x, y) and $(x + 1, y)$, and another bi-directional trail connecting points (x, y) and $(x, y + 1)$. What's more, there are n additional bi-directional trails, where the i-th trail connects points (x_i, y_i) and $(x_i + 1, y_i + 1)$.

BaoBao can only move along the trails. Let $f(x, y)$ be the smallest number of trails BaoBao has to pass if he wants to move from point $(0, 0)$ to point (x, y). Given two integers p and q, calculate

$$\sum_{x=0}^{p}\sum_{y=0}^{q} f(x, y)$$

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 three integers n, p, and q ($1 \le n \le 10^6$, $0 \le p, q \le 10^6$). Their meanings are described above.

For the following n lines, the i-th line contains two integers x_i and y_i ($0 \le x_i, y_i \le 10^6$) indicating that the i-th additional trail connects points (x_i, y_i) and $(x_i + 1, y_i + 1)$. It's guaranteed that $x_i \neq x_j$ or $y_i \neq y_j$ for all $i \neq j$.

It's guaranteed that the sum of n of all test cases does not exceed 10^6. Note that there is no constraint on the sum of p or q.

Output

For each test case output one line containing one integer indicating the answer.

Example

standard input	standard output
2 3 2 4 1 1 0 2 0 0 1 100 100 1000 1000	34 1020100

Problem M. Italian Cuisine

BaoBao has prepared a pizza for you! This pizza is a convex polygon with stuffed crust along all of its edges. However, the crust is delicate, allowing you to cut only through its vertices and not into the middle of the edges. Unfortunately, there's a sizable circular piece of pineapple on the pizza that you absolutely want to avoid.

Calculate the largest single piece of pineapple-free pizza that you can obtain with a single straight cut and output its size. A piece is considered pineapple-free when no part of the pineapple falls strictly within it. That is, the area of intersection of the pineapple and the pizza is 0.

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 ($3 \le n \le 10^5$) indicating the number of vertices of the pizza.

The second line contains three integers x_c, y_c, and r ($-10^9 \le x_c, y_c \le 10^9$, $1 \le r \le 10^9$), indicating the coordinates of the center of the pineapple and its radius.

For the following n lines, the i-th line contains two integers x_i and y_i ($-10^9 \le x_i, y_i \le 10^9$) indicating the coordinate of the i-th vertex. The vertices are listed in counter-clockwise order. No two points coincide. However there might be three points lying on the same line.

It is guaranteed that no part of the pineapple lies outside of the boundaries of the pizza. It's also guaranteed that the sum of n of all the test cases does not exceed 10^5.

Output

For each test case, output one line containing one integer, indicating the size of the largest piece of pineapple-free pizza multiplied by 2. It can be proven that this value will always be an integer. If there's no way to get a pineapple-free piece, output 0.

Example

standard input	standard output
3 5 1 1 1 0 0 1 0 5 0 3 3 0 5 6 2 4 1 2 0 4 0 6 3 4 6 2 6 0 3 4 3 3 1 3 0 6 3 3 6 0 3	5 24 0

Note

The sample test cases are shown below.