The 2026 Shandong Provincial Collegiate Programming Contest

Contest Session

May 24, 2026

Problem List

A	Klotski
B	Dictionary 2
C	Meeting Schedule
D	Largest Digit 2
E	Simple Constructive Problem
F	Gifts in Place
G	Vampire Crawlers
H	Puzzle
I	Version Number
J	Making Pine Branches
K	Minimum Spanning Tree
L	Fraction Iteration
M	Night at the Museum 2

This problem set should contain 13 (thirteen) problems on 18 (eighteen) 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. Klotski

There is a grid with n rows and m columns. In each cell, there is an arrow pointing either up (`U`), down (`D`), left (`L`), or right (`R`).

In each operation, you may remove exactly one arrow. However, to remove an arrow, there must be no other arrows in the direction it points. More specifically, let $s_{i,j}$ be the arrow located in row i and column j. To remove $s_{i,j}$:

- If $s_{i,j} = \texttt{U}$, all $s_{k,j}$ satisfying $1 \le k < i$ must be removed beforehand.
- If $s_{i,j} = \texttt{D}$, all $s_{k,j}$ satisfying $i < k \le n$ must be removed beforehand.
- If $s_{i,j} = \texttt{L}$, all $s_{i,k}$ satisfying $1 \le k < j$ must be removed beforehand.
- If $s_{i,j} = \texttt{R}$, all $s_{i,k}$ satisfying $j < k \le m$ must be removed beforehand.

For each arrow, answer the following question separately: What is the minimum number of operations needed to remove it?

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 two integers n and m ($1 \le n, m \le 5 \times 10^4$, $n \times m \le 5 \times 10^4$), indicating the number of rows and columns in the grid.

In the following n lines, the i-th line contains a string $s_{i,1}s_{i,2}\ldots s_{i,m}$ of length m ($s_{i,j} \in \{\texttt{U}, \texttt{D}, \texttt{L}, \texttt{R}\}$), where $s_{i,j}$ indicates the arrow in the i-th row and j-th column.

It is guaranteed that the sum of $n \times m$ over all test cases does not exceed 5×10^4.

Output

For each test case, output n lines. The i-th line contains m integers $a_{i,1}, a_{i,2}, \ldots, a_{i,m}$, separated by spaces, where $a_{i,j}$ is the minimum number of operations needed to remove the arrow $s_{i,j}$. If it is impossible to remove the arrow $s_{i,j}$, then $a_{i,j}$ should be -1.

Example

standard input	standard output
2 4 3 RRD DUL RDD RRR 1 2 LR	-1 -1 -1 7 -1 -1 5 3 2 3 2 1 1 1

Problem B. Dictionary 2

You are given a rooted tree with n vertices, where the root is vertex 1. Each vertex u is assigned a lowercase English character c_u. For any vertex u, let $S(u)$ denote the string formed by concatenating the characters on the path from u up to the root. More formally:

- If $u = 1$, then $S(u) = c_u$.
- If v is a child of u, then $S(v) = c_v + S(u)$, where $+$ denotes string concatenation.

Let $\mathcal{V}$ be the set of all vertices in the tree. Let $\mathcal{D}$ be the set of all nonempty substrings of the strings in $\{S(u) \mid u \in \mathcal{V}\}$. That is, $w \in \mathcal{D}$ if and only if w is a nonempty substring of $S(u)$ for some vertex u. Recall that sets do not contain duplicate elements.

We define the lexicographical order $\leq_{lex}$ as follows: for two strings A and B, $A \leq_{lex} B$ if and only if A is a prefix of B or, at the first position where they differ, the character in A is smaller than the corresponding character in B.

You need to process q operations sequentially. For each operation, you are given an encoded integer u' and a character c. Let N be the number of vertices before this operation, and let l be the integer output by the previous operation (for the first operation, $l = 0$). Perform the following steps:

1. Calculate the real parent vertex $u = ((u' + l) \bmod N) + 1$. Create a new vertex with index $(N + 1)$ as a child of u with character c, forming the new string $S(N + 1) = c + S(u)$. Update the set $\mathcal{D}$ to include all nonempty substrings of $S(N + 1)$ that were not previously present in $\mathcal{D}$.
2. Output the number of strings in the updated set $\mathcal{D}$ that are lexicographically less than or equal to $S(N + 1)$. More formally, output the value:

$$|\{w \in \mathcal{D} \mid w \leq_{lex} S(N + 1)\}|$$

Input

There is only one test case in each test file.

The first line contains two integers n and q ($1 \leq n, q \leq 10^5$).

The second line contains a string $c_1c_2 \ldots c_n$ consisting of lowercase English letters, where c_i is the character of vertex i.

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

For the following q lines, the i-th line contains an integer u'_i ($1 \leq u'_i \leq N$, where N is the number of vertices before this operation) and a lowercase English letter c_i, indicating the i-th operation.

Output

For each operation, output one line containing an integer, indicating the answer.

Example

standard input	standard output
2 3 ab 1 2 1 b 1 a 2 c	5 2 8

Note

We explain the sample test case below.

Initially, $S(1) = \mathrm{a}$, $S(2) = \mathrm{ba}$, and $\mathcal{D} = \{\mathrm{a}, \mathrm{b}, \mathrm{ba}\}$.

1. Input $u' = 1, c = \mathrm{b}$. The decrypted parent is $u = 2$. Create vertex 3, $S(3) = \mathrm{bba}$. Added substrings $\mathrm{bb}, \mathrm{bba}$. There are 5 strings in $\mathcal{D}$ with $\leq_{\mathrm{lex}} \mathrm{bba}$.
2. Input $u' = 1, c = \mathrm{a}$. The decrypted parent is $u = 1$. Create vertex 4, $S(4) = \mathrm{aa}$. Added substring aa. There are 2 strings in $\mathcal{D}$ with $\leq_{\mathrm{lex}} \mathrm{aa}$.
3. Input $u' = 2, c = \mathrm{c}$. The decrypted parent is $u = 1$. Create vertex 5, $S(5) = \mathrm{ca}$. Added substrings c, ca. There are 8 strings in $\mathcal{D}$ with $\leq_{\mathrm{lex}} \mathrm{ca}$.

Problem C. Meeting Schedule

A company has n employees, and each employee i has a free time window $[l_i, r_i]$ (both endpoints are integers) during which they can attend meetings.

The company is organizing a team-building event where every pair of employees needs to have a one-on-one conversation. For two employees to meet, their free time windows must intersect (share at least one common point).

Employees can adjust their schedules by changing the endpoints of their free time windows. However, rescheduling is troublesome, and changing an endpoint from time a to time b causes $(a-b)^2$ units of frustration. Larger adjustments require moving more appointments around, so the frustration grows quadratically. After adjustment, each time window must remain valid (the left endpoint smaller than or equal to the right endpoint, and both endpoints are still integers).

You need to find the minimum total frustration so that every pair of employees can find a time to meet.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 10^4$), indicating the number of test cases. For each test case:

The first line contains an integer n ($2 \le n \le 2 \times 10^5$), indicating the number of employees.

For the following n lines, the i-th line contains two integers l_i and r_i ($0 \le l_i \le r_i \le 10^6$), indicating the free time window of the i-th employee.

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

Output

For each test case, output one line containing an integer, indicating the minimum total frustration.

Example

standard input	standard output
3 3 6 8 0 2 1 7 2 1 3 2 4 2 1 1 100 100	8 0 4901

Note

For the first sample test case, employee 1 can adjust his/her window from $[6, 8]$ to $[4, 8]$ with frustration $(6-4)^2 = 4$, and employee 2 can adjust from $[0, 2]$ to $[0, 4]$ with frustration $(4-2)^2 = 4$. Now the three windows $[4, 8]$, $[0, 4]$, $[1, 7]$ pairwise intersect, so every pair can meet. The total frustration is $4 + 4 = 8$.

For the second sample test case, the two employees' windows already intersect, so no adjustment is needed.

Problem D. Largest Digit 2

Let $f(x)$ be the largest digit in the decimal representation of a positive integer x. For example, $f(4523) = 5$ and $f(1001) = 1$.

Given four positive integers l_a, r_a, l_b, and r_b such that $l_a \le r_a$ and $l_b \le r_b$, you need to calculate

$$\sum_{a=l_a}^{r_a} \sum_{b=l_b}^{r_b} f(a+b)$$

That is, the sum of $f(a+b)$ over all (a, b) that satisfy $l_a \le a \le r_a$ and $l_b \le b \le r_b$.

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 and only line contains four integers l_a, r_a, l_b, and r_b ($1 \le l_a \le r_a \le 10^9$, $1 \le l_b \le r_b \le 10^9$).

Output

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

Examples

standard input	standard output
2 178 182 83 85 2 5 3 6	91 100
1 1 1000000000 1 1000000000	8425695016000000001

Problem E. Simple Constructive Problem

Given a grid with n rows and m columns, fill each cell with an integer from 0 to $(nm-1)$ (using each integer exactly once) such that for all $0 \le i < nm$, the cell containing i and the cell containing $(i+1) \bmod nm$ are not adjacent (i.e., they do not share an edge).

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 10^4$), indicating the number of test cases. For each test case:

The first line contains two integers n and m ($1 \le n, m \le 2 \times 10^5$, $3 \le n \times m \le 2 \times 10^5$), indicating the number of rows and columns in the grid.

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

Output

For each test case:

- If a valid arrangement exists, first output `Yes` in one line. Then output n lines. The i-th line contains m integers separated by a space, where the j-th integer represents the number filled in the cell at row i and column j. If there are multiple valid arrangements, output any one of them.
- Otherwise, if there is no valid arrangement, just output `No` in one line.

Example

standard input	standard output
2 3 4 1 3	Yes 4 9 7 1 10 0 5 3 6 2 11 8 No

Problem F. Gifts in Place

There are n people sitting around a round table, numbered from 1 to n in clockwise order. For each $i = 1, 2, \ldots, n$, the i-th person is adjacent to the $((i \bmod n) + 1)$-th person.

There are n gifts, numbered from 1 to n. Initially, for each $i = 1, 2, \ldots, n$, the i-th person receives the p_i-th gift. However, due to a hasty distribution, at least one person ends up with the wrong gift. Under the correct plan, for each $i = 1, 2, \ldots, n$, the i-th person should receive the q_i-th gift.

You can perform operations where each operation allows swapping the gifts held by any two adjacent people. Find the minimum number of operations required so that everyone receives the correct gift.

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 an integer n ($3 \le n \le 3 \times 10^3$).

The second line contains n distinct integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$).

The third line contains n distinct integers $q_1, q_2, \ldots, q_n$ ($1 \le q_i \le n$).

It is guaranteed that the sum of n over all test cases does not exceed 3×10^3.

Output

For each test case, output two lines:

The first line contains an integer k, indicating the minimum number of operations.

The second line contains k integers $x_1, x_2, \ldots, x_k$, where the i-th integer x_i indicates that, in the i-th operation, you swap the gifts held by the x_i-th person and the $((x_i \bmod n) + 1)$-th person.

If there are multiple valid solutions, you may output any of them.

Example

standard input	standard output
2 3 1 2 3 3 2 1 4 3 4 1 2 1 2 3 4	1 3 4 2 3 1 2

Problem G. Vampire Crawlers

Recently, the roguelike deckbuilder "Vampire Crawlers", developed by poncle, has been officially released. In the game, players build decks and play cards during combat to deal damage. The core mechanic is the combo system: when you play cards in ascending order of mana cost, the damage multiplier keeps stacking.

You currently have n cards, where the i-th card has a cost and a base damage. You need to choose an order to play all n cards one by one. When you play a card:

- If this is NOT the first card played, and its cost is equal to that of the previous card plus 1, the damage multiplier is increased by 1.
- Otherwise, the damage multiplier is reset to 1.

After updating the damage multiplier, the damage dealt by this card is calculated as its base damage times the multiplier.

You need to find the maximum possible total damage.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 10^4$), indicating the number of test cases. For each test case:

The first line contains an integer n ($1 \le n \le 2 \times 10^5$), indicating the number of cards.

For the following n lines, the i-th line contains two integers a_i and b_i ($0 \le a_i \le 10^9$, $1 \le b_i \le 10^6$), indicating the cost and damage of the i-th card.

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

Output

For each test case, output one line containing an integer, indicating the maximum total damage.

Example

standard input	standard output
2 9 1 3 3 10 6 4 1 2 5 8 0 5 2 7 6 1 2 7 1 1000000000 1000000	105 1000000

Note

For the first sample test case, play the cards in the order of $6, 1, 9, 2, 8, 4, 7, 5, 3$. The total damage is $5 \times 1 + 3 \times 2 + 7 \times 3 + 10 \times 4 + 1 \times 1 + 2 \times 1 + 7 \times 2 + 8 \times 1 + 4 \times 2 = 105$.

Problem H. Puzzle

There is a grid with n rows and m columns. Each cell is either empty (denoted by `.`) or blocked (denoted by `#`).

You are also given k puzzle pieces, numbered $1, 2, \ldots, k$. Each piece is a 4-connected polyomino that cannot be rotated or flipped. Each piece is described by its minimum bounding rectangle, where `#` denotes a cell belonging to the piece and `.` denotes a cell not belonging to the piece.

You need to select some of the k pieces and place them on the grid such that:

- Each cell of each selected piece must cover exactly one empty cell of the grid.
- No two selected pieces cover the same cell.
- Every empty cell of the grid is covered by exactly one selected piece.

Count the number of valid arrangements. Two arrangements are considered different if and only if they use different sets of puzzle pieces, or there exists a piece whose covered cells differ between the two arrangements.

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, m, and k ($1 \le n, m \le 8$, $1 \le k \le 8$), indicating the number of rows, columns, and puzzle pieces.

The next n lines each contain a string of length m, describing the grid. Here `.` denotes an empty cell and `#` denotes a blocked cell. It's guaranteed that the grid contains at least one empty cell.

Then k pieces are described. For each piece, the first line contains two integers r and c ($1 \le r, c \le 8$), indicating the number of rows and columns of its minimum bounding rectangle. The next r lines each contain a string of length c, describing the piece. It is guaranteed that each piece is a 4-connected polyomino, and that every row and every column of the bounding rectangle contains at least one `#`.

It is guaranteed that at most 10 test cases satisfy k greater than 5.

Output

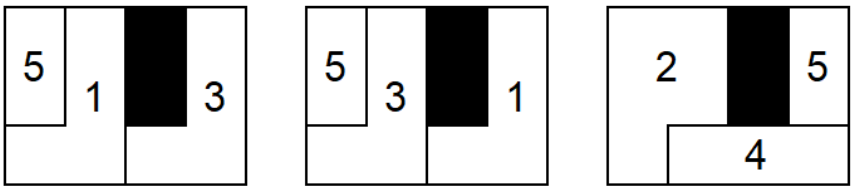
For each test case, output one line containing an integer, indicating the number of valid arrangements.

Example

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

Note

The 3 arrangements of the first sample test case are described below, where the integers indicate the indices of the pieces.



Problem I. Version Number

In the early days of large language models, researchers discovered an interesting phenomenon: when asked "Which is larger, 5.11 or 5.9", many models would confidently answer 5.11. It is suspected that the models confused the comparison rules for decimal numbers with those for version numbers — in version number comparison, 5.11 is indeed greater than 5.9, since the minor version 11 is greater than 9.

Each version number consists of two non-negative integers: a major version x and a minor version y, denoted as $x.y$.

The comparison rule for version numbers is as follows: first compare the major versions; the version with the larger major version is greater. If the major versions are equal, compare the minor versions; the version with the larger minor version is greater. Formally, version $x_1.y_1$ is greater than version $x_2.y_2$ if and only if $x_1 > x_2$, or $x_1 = x_2$ and $y_1 > y_2$.

Given n version numbers, you need to find the largest one among them.

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 an integer n ($2 \le n \le 100$), indicating the number of version numbers.

For the following n lines, the i-th line contains two integers x_i and y_i ($0 \le x_i, y_i \le 10^3$), indicating the version number $x_i.y_i$.

Output

For each test case, output one line containing two integers x' and y' separated by a space, indicating the major version and the minor version of the largest version number.

Example

standard input	standard output
3 3 5 11 5 9 3 12 2 1 0 0 1 2 0 0 0 0	5 11 1 0 0 0

Problem J. Making Pine Branches

Workers in the artificial pine branch factory need to insert pine needles of different sizes onto branches to make various kinds of decorations.

There is an empty branch and two boxes on the table. One of the boxes is empty, while the other contains n pieces of pine needles stacked in a pile. The size of the i-th piece of pine needle, from top to bottom, is a_i. Each time, the worker can perform one of the following two operations:

- Choose a box containing pine needles and move the pine needle at the top of the box to the top of the other box.
- Choose a box containing pine needles and insert the pine needle at the top of the box onto the branch.

The worker has to insert all n pieces of pine needles onto the branch. Also, when inserting a piece of pine needle, its size must not be larger than that of the previously inserted piece. That is to say, let b_i be the size of the i-th piece of pine needle inserted onto the branch; then $b_1 \geq b_2 \geq \ldots \geq b_n$ must hold.

You need to find the smallest possible number of operations performed so that all pine needles can be inserted onto the branch.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \leq T \leq 10^4$), indicating the number of test cases. For each test case:

The first line contains an integer n ($1 \leq n \leq 4 \times 10^5$), indicating the number of pieces of pine needles.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$), where a_i indicates the size of the i-th piece of pine needle, from top to bottom, in the box.

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

Output

For each test case, output one line containing an integer, indicating the smallest possible number of operations.

Example

standard input	standard output
2 5 7 3 10 7 2 3 1 1 1	8 3

Note

We show an optimal strategy for the first sample test case below. The sizes of pine needles in the boxes are listed from top to bottom, and the sizes of pine needles on the branch are listed in insertion order.

#	Box A	Box B	Branch		#	Box A	Box B	Branch
1	$\{3, 10, 7, 2\}$	$\{7\}$	$\{\}$		5	$\{3, 2\}$	$\{7\}$	$\{10, 7\}$
2	$\{10, 7, 2\}$	$\{3, 7\}$	$\{\}$		6	$\{3, 2\}$	$\{\}$	$\{10, 7, 7\}$
3	$\{7, 2\}$	$\{3, 7\}$	$\{10\}$		7	$\{2\}$	$\{\}$	$\{10, 7, 7, 3\}$
4	$\{2\}$	$\{3, 7\}$	$\{10, 7\}$		8	$\{\}$	$\{\}$	$\{10, 7, 7, 3, 2\}$

Problem K. Minimum Spanning Tree

Given an undirected connected graph with n vertices and m weighted edges, where k vertices are marked special, we say a spanning tree is good if all special vertices are leaves in the tree.

You need to find the minimum total weight of all edges among all good spanning trees.

Recall that a spanning tree is a connected subgraph of the original graph containing n vertices and $(n-1)$ edges. Also, recall that leaves in a tree are the vertices connected with exactly one edge.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 10^4$), indicating the number of test cases. For each test case:

The first line contains three integers n, m, and k ($2 \le n \le 2 \times 10^5$, $n - 1 \le m \le 2 \times 10^5$, $1 \le k \le n$), indicating the number of vertices, edges, and special vertices in the graph.

The second line contains k distinct integers $a_1, a_2, \ldots, a_k$ ($1 \le a_i \le n$), indicating the special vertices.

For the following m lines, the i-th line contains three integers u_i, v_i, and w_i ($1 \le u_i, v_i \le n$, $1 \le w_i \le 10^9$), indicating that there is an edge connecting vertex u_i and v_i with weight w_i. It's guaranteed that the given graph is connected, but there may be self loops or multiple edges.

It is guaranteed that both the sum of n and the sum of m over all test cases do not exceed 2×10^5.

Output

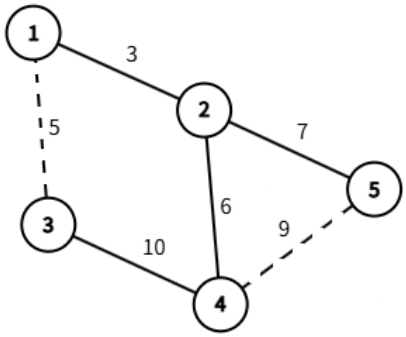
For each test case, output one line containing an integer, indicating the minimum total weight of all edges in a good spanning tree. If there is no good spanning tree, output `-1` instead.

Example

standard input	standard output
3 5 6 2 1 5 1 2 3 2 5 7 4 2 6 5 4 9 3 4 10 1 3 5 4 4 4 1 2 3 4 1 2 1 2 3 1 3 4 1 4 1 1 3 4 1 1 1 2 10 1 2 100 2 3 1000 3 3 100000	26 -1 1010

Note

The first sample test case is illustrated below, where the solid lines are the edges in the spanning tree, and the dashed lines are the edges not in the tree.


1
2
3
4
5
3
5
7
6
9
10

Problem L. Fraction Iteration

You are given an irreducible fraction $\frac{a}{b}$ (i.e., $\gcd(a, b) = 1$). Every second, both the numerator and the denominator increase by 1, and then the fraction is immediately reduced to its simplest form.

Formally, if the current fraction is $\frac{a}{b}$, after one second it becomes $\frac{a+1}{b+1}$, then let $g = \gcd(a + 1, b + 1)$, and update it to $\frac{(a+1)/g}{(b+1)/g}$.

Given the initial fraction $\frac{a}{b}$ and q queries, each providing an integer k_i, you need to find the fraction exactly after k_i seconds, starting from the initial state.

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 a, b, and q ($1 \le a, b \le 10^{12}$, $1 \le q \le 100$). It is guaranteed that $\gcd(a, b) = 1$ initially.

For the following q lines, the i-th line contains an integer k_i ($1 \le k_i \le 10^{18}$), indicating the time of the query.

Output

For each query, output one line containing two integers a' and b' separated by a space, indicating the numerator and denominator of the fraction exactly after k_i seconds, starting from the initial state.

Example

standard input	standard output
3 1 7 3 1 2 3 1 1 1 1 13 3 3 7 2 10	1 4 2 5 1 2 1 1 5 4 8 3 8 7

Problem M. Night at the Museum 2

A security guard patrols a museum to monitor exhibits. The patrol route is a closed polyline consisting of n points labeled from 1 to n, which are connected in order (the last point connects back to the first point) to form n consecutive segments. The guard patrols in the order $1 \to 2 \to \cdots \to n \to 1 \to \cdots$. When the guard is not at a segment endpoint, their field of view is a sector with radius $10^{10^{10^{10}}}$ and central angle $2a$, with the sector's bisector aligned with the forward direction of the patrol segment.

A point P on a patrol segment (endpoints exclusive) can see an exhibit at point Q if and only if Q lies within the sector region with apex at P. Given the closed polyline patrol route defined by n points and m exhibit locations, find the total length of patrol routes where all exhibits can be seen at the same time.

Input

There is only one test case in each test file.

The first line contains three integers n, m, and a ($3 \le n \le 2 \times 10^5$, $1 \le m \le 2 \times 10^5$, $0 < a < 90$), indicating the number of points of the closed polyline, the number of exhibits, and the half-angle of the field of view in degrees, respectively.

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 coordinates of the i-th point of the polyline. Consecutive points form patrol segments, and the last point connects back to the first point to form a closed loop. It is guaranteed that no two consecutive points (including the last and first points) are identical.

For the following m 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 coordinates of the i-th exhibit.

Output

Output one line containing a real number, indicating the total length of patrol routes where all exhibits can be seen at the same time.

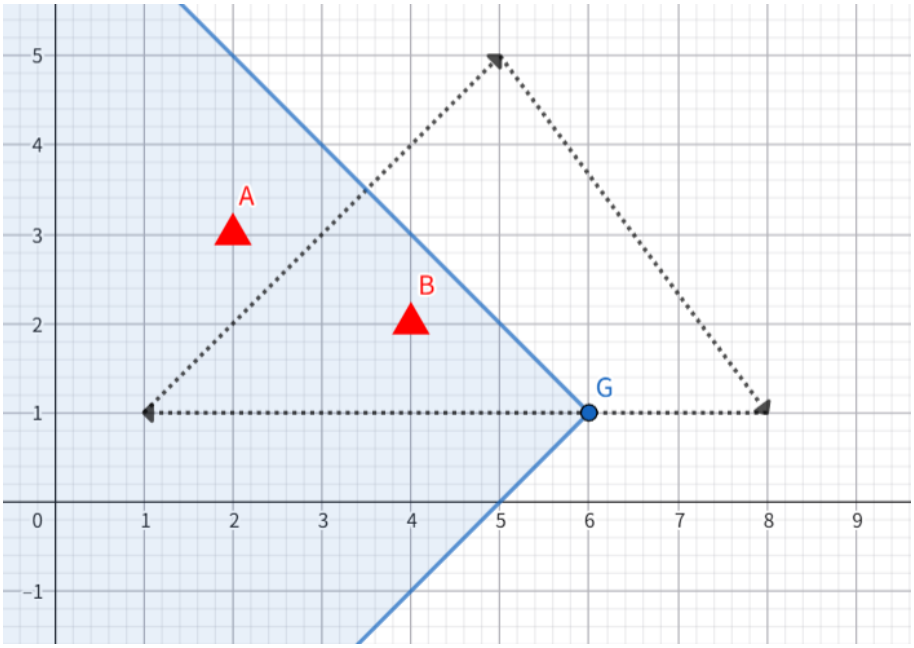
Your answer will be considered correct if its absolute or relative error does not exceed 10^{-6}. Formally speaking, suppose that your answer is a and the jury's answer is b, your answer is accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$.

Examples

standard input	standard output
3 2 45 1 1 5 5 8 1 2 3 4 2	4.4142135624
4 1 60 0 0 2 0 2 2 0 2 1 1	1.6905989232

Note

The first sample test case is illustrated below. A and B are the two exhibits, and G is the guard.


A
B
G
5
4
3
2
1
0
−1
1
2
3
4
5
6
7
8
9