Problem A. AUS

Pigeland University will host the 2224 Animal Collegiate Programming Contest (ACPC 2224). Unlike previous years, where teams from the host institution were marked as unofficial, Pigeland will send official teams to compete in the contest with three workstations for each team. Nevertheless, Pig-head, the coach of Pigeland University, remains uncertain about his teams' chances of securing gold medals. As a result, he decides to acquire the contest problems in advance from the AUS problem-setting group, using the excuse of needing to upload data to the online judge.

To prevent cheating, AUS attempts to encrypt the problems using a special cipher. Specifically, problems are represented by strings consisting of lowercase English letters. AUS wants to design a cipher function $f(x)$ that maps lowercase English letters to lowercase English letters. For a problem $S = s_1 s_2 \dots s_n$, the encrypted version of the problem is another string given by $F(S) = f(s_1)f(s_2)\dots f(s_n)$. For example, when $S = \texttt{abcabc}$ and $f(\texttt{a}) = \texttt{a}$, $f(\texttt{b}) = \texttt{k}$, $f(\texttt{c}) = \texttt{a}$, the encrypted version is $F(S) = \texttt{akaaka}$.

As a member of AUS, your task is to design the cipher function f. The leader of AUS believes that the function is *strong* if and only if there exists at least one problem that can be encrypted into the same encrypted version as another, while not all problems produce the same encrypted output. To validate this, he will give you three problems S_1, S_2, and S_3, and you need to find a cipher function f such that $F(S_1) = F(S_2)$ and $F(S_1) \neq F(S_3)$. Since AUS has several experienced members, your task is simply to determine whether such a cipher function exists.

Input

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

The first line contains a string S_1 ($1 \le |S_1| \le 10^3$) consisting of only lowercase English letters.

The second line contains a string S_2 ($1 \le |S_2| \le 10^3$) consisting of only lowercase English letters.

The third line contains a string S_3 ($1 \le |S_3| \le 10^3$) consisting of only lowercase English letters.

It is guaranteed that the sum of $|S_1| + |S_2| + |S_3|$ of all test cases does not exceed 3×10^4.

Output

For each test case, if such cipher function exists, output `YES` in one line. Otherwise, output `NO` instead.

Example

standard input	standard output
4 abab cdcd abce abab cdcd abcd abab cdcd abc x yz def	YES NO YES NO

Note

For the first and third sample test cases, one valid cipher function can be $f(\texttt{a}) = f(\texttt{b}) = f(\texttt{c}) = f(\texttt{d}) = \texttt{a}$ and $f(\texttt{e}) = \texttt{b}$.

Problem B. Barkley III

There are n little pigs in Pigeland. All of them are proficient in competitive programming, and the i-th of them has a_i rating. If k pigs $p_1, p_2, \cdots, p_k$ form a team, the rating of the team will be a_{p_1} & a_{p_2} & a_{p_3} & $\cdots$ & a_{p_k}, where & denotes the bitwise AND operation.

There are some programming contests to take place. Pigeland can send exactly one team to participate in each contest. For the i-th competition, only the pigs numbered between l_i and r_i (both inclusive) have time to participate. Unfortunately, due to a shortage of funds, exactly one pig numbered between l_i and r_i has to be removed. Meanwhile, all other pigs in the interval will participate in the contest. Pig-head, the coach of Pigeland, needs to properly select pigs who will not participate so that the team's rating is maximized.

However, through training and participating in contests, the rating of pigs may be changed. As Pig-head's best friend, your task is to maintain the pigs' rating for the following q events belonging to three types.

- 1 l r x: Pig-head changes the rating of each pig numbered between l and r (both inclusive) by executing the bitwise AND operation with x. More formally, for all $l \le i \le r$, a_i becomes a_i & x.
- 2 s x: Pig-head changes the rating of the s-th pig to x.
- 3 l r: Pig-head asks for the maximum rating when forming a team by selecting pigs numbered between l and r (both inclusive) and removing exactly one of them.

Input

There is only one test case in each test file.

The first line contains two integers n and q ($2 \le n \le 10^6$, $1 \le q \le 10^6$) indicating the number of pigs and the number of events.

The second line contains n integers $a_1, a_2, \cdots, a_n$ ($0 \le a_i < 2^{63}$) where a_i indicates the rating of the i-th pig.

For the following q lines, the i-th line first contains an integer op_i ($op_i \in \{1, 2, 3\}$) indicating the type of the i-th event. If $op_i = 1$, then three integers l, r, and x follow ($1 \le l \le r \le n$, $0 \le x < 2^{63}$); If $op_i = 2$, then two integers s and x follow ($1 \le s \le n$, $0 \le x < 2^{63}$); If $op_i = 3$, then two integers l and r follow ($1 \le l < r \le n$).

Output

For each event of the third type, output one line containing one integer indicating the maximum rating of the team.

Example

standard input	standard output
5 9 7 7 7 6 7 3 1 5 2 1 3 3 1 5 3 1 3 1 1 2 3 3 1 3 2 2 8 3 1 3 3 1 2	7 6 7 3 3 8

Problem C. Catch the Star

BaoBao bought a telescope to observe a star in the night sky. The star is represented as a convex polygon S. However, there are n convex polygonal moons M_i that could obstruct his view. BaoBao can place his telescope anywhere on the x-axis between points $(l, 0)$ and $(r, 0)$, but he **cannot** place it exactly at $(l, 0)$ or $(r, 0)$.

Your task is to help BaoBao find the total length of the segments on the x-axis where he can position his telescope such that he has an unobstructed view of the star S, and whether he **can** find a position at all. An unobstructed view means that no line segment from the chosen point on the x-axis to any point inside or on S properly intersects any of the moons M_i. The line segment is allowed to touch the boundary of the moons but not cross through them.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 2.5 \times 10^4$) indicating the number of test cases. For each test case:

The first line contains three integers n, l, and r ($1 \le n \le 10^4$, $-10^9 \le l < r \le 10^9$), denoting the number of moons and the range for telescope placement.

The second line describes the star S and begins with an integer k_0 ($3 \le k_0 \le 10^5$), denoting the number of vertices of S, followed by $2 \times k_0$ integers $x_{0,1}, y_{0,1}, x_{0,2}, y_{0,2}, \cdots, x_{0,k_0}, y_{0,k_0}$ ($-10^9 \le x_{0,j}, y_{0,j} \le 10^9$), where $(x_{0,j}, y_{0,j})$ is the coordinate of the j-th vertex of S in counter-clockwise order.

For the following n lines, the i-th line describe moon M_i. Each line begins with an integer k_i ($3 \le k_i \le 10^5$), denoting the number of vertices of M_i, followed by $2 \times k_i$ integers $x_{i,1}, y_{i,1}, x_{i,2}, y_{i,2}, \cdots, x_{i,k_i}, y_{i,k_i}$ ($-10^9 \le x_{i,j}, y_{i,j} \le 10^9$), where $(x_{i,j}, y_{i,j})$ is the coordinate of the j-th vertex of M_i in counter-clockwise order.

It is guaranteed that S and M_i are convex polygons. The star S, the moons M_i, and the segment from $(l, 0)$ to $(r, 0)$ do not touch or intersect with each other. However, different moons M_i can intersect with each other. No three consecutive vertices of the same polygon are collinear.

It's guaranteed that the sum of $\sum_{i=0}^{n} k_i$ of all test cases does not exceed 10^6. It is also guaranteed that the total number of polygons of all test cases (which is $\sum(n+1)$) does not exceed 5×10^4.

Output

For each test case, output a single line containing the total length of valid segments on the x-axis, strictly between $(l, 0)$ and $(r, 0)$, where BaoBao can place his telescope to see S without obstruction. If there's no valid point between $(l, 0)$ and $(r, 0)$, output -1 instead.

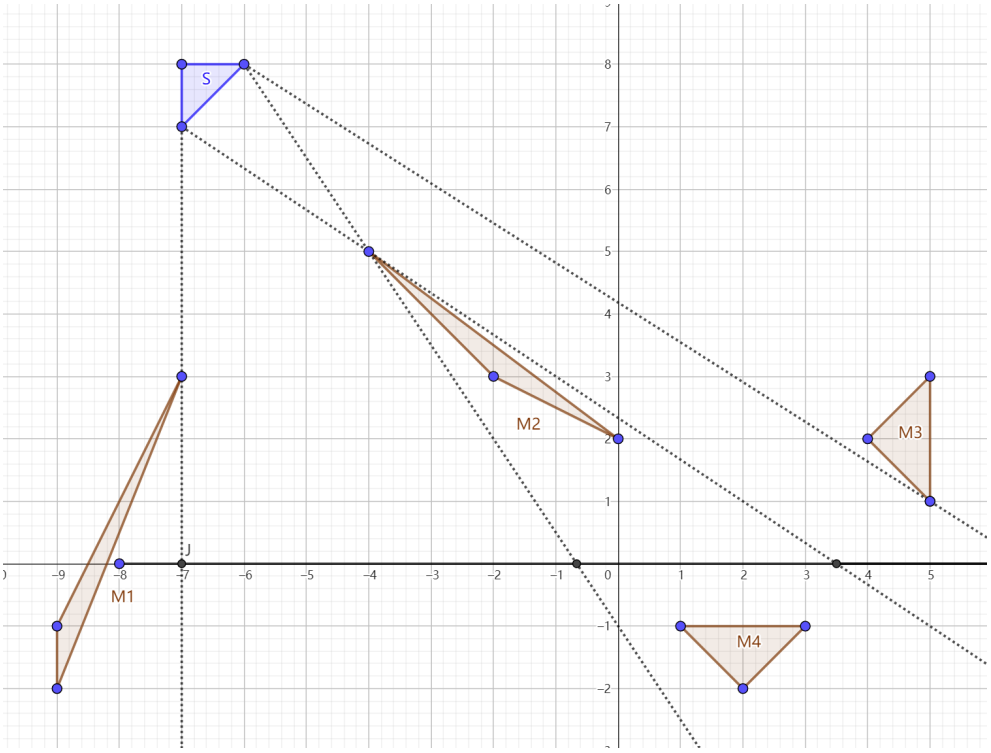
Your answer would be considered correct if the relative or absolute error does not exceed 10^{-9}.

Examples

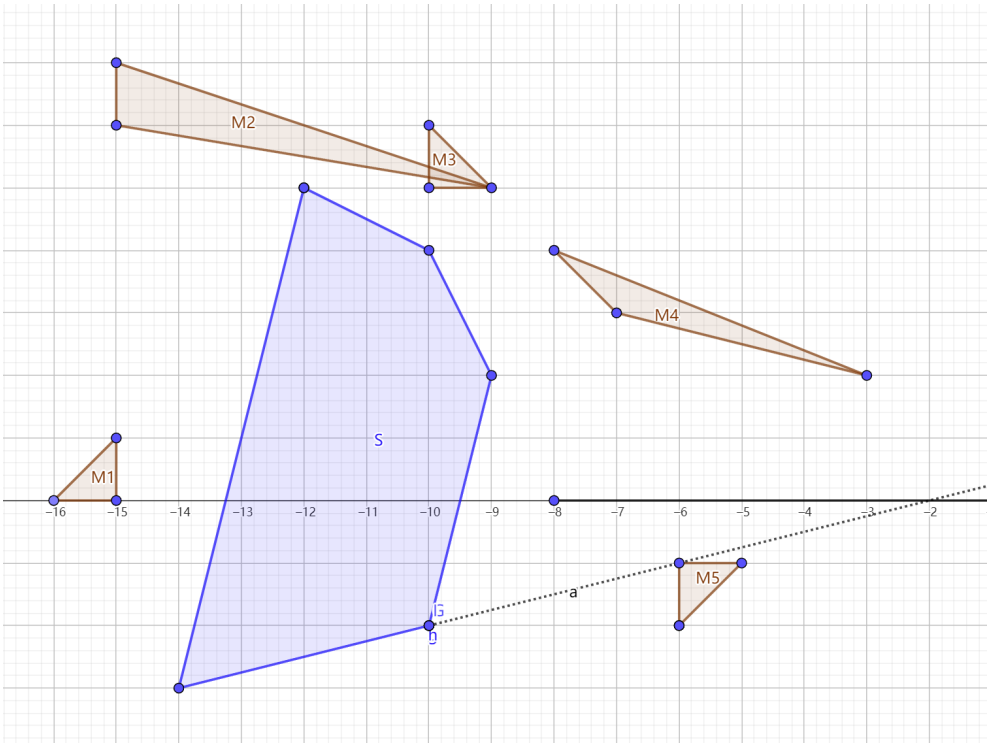
standard input	standard output
2 4 -8 8 3 -7 7 -6 8 -7 8 3 -9 -2 -7 3 -9 -1 3 -2 3 0 2 -4 5 3 5 1 5 3 4 2 3 1 -1 2 -2 3 -1 5 -8 8 5 -14 -3 -10 -2 -9 2 -10 4 -12 5 3 -16 0 -15 0 -15 1 3 -15 6 -9 5 -15 7 3 -10 5 -9 5 -10 6 3 -7 3 -3 2 -8 4 3 -6 -1 -6 -2 -5 -1	9.404761904761905 6.000000000000000
3 1 -4 4 3 -2 6 0 5 2 6 3 -3 1 3 1 0 4 3 -2 2 3 -2 4 2 4 0 6 3 -2 2 -1 2 -2 3 3 1 2 2 2 2 3 3 -2 -1 0 -3 2 -1 1 1 2 3 -8 0 -7 0 -8 1 3 -5 0 -4 -1 -4 0	-1.000000000000000 0.000000000000000 1.000000000000000
1 1 -744567334 955216804 5 -781518205 -852078097 -781516900 -852078384 -781516392 -852076569 -781518329 -852076047 -781519925 -852077600 5 -393011614 -131855702 -393010699 -131856607 -393008846 -131856475 -393009388 -131854587 -393010201 -131854694	1699779738.691979192313738

Note

For the first sample, the first test case is illustrated below; the telescope can be located between $(-7, 0)$ and $(-\frac{2}{3}, 0)$, or $(\frac{7}{2}, 0)$ and $(\frac{46}{7}, 0)$.



The second test case is illustrated below; the telescope can be located between $(-8, 0)$ and $(-2, 0)$.



Note that the input format in the third sample is for presentation purposes only, because we cannot print all the coordinates in one line without exceeding the width of the paper. Please refer to the other samples for more accurate input formatting.

Problem D. Dividing Sequence

Alice got a sequence A constructed by her neighbors. Since Alice doesn't like long sequences, she decides to divide the sequence into two (possibly empty) sequences B and C and give them back to her neighbors. Her division should meet the following constraints:

- B and C are both subsequences of sequence A.
- Each element of A belongs to exactly one of the sequences B or C.
- $B \leq C$ in lexicographical order.

Here we define a sequence $P = p_1, p_2, \cdots, p_u$ of length u to be lexicographically smaller than a sequence $Q = q_1, q_2, \cdots, q_v$ of length v if one of the following constraints is true:

- $u < v$ and P is a prefix of Q.
- There exists an integer $1 \leq k \leq \min(u, v)$ such that $p_i = q_i$ for all $1 \leq i < k$ and $p_k < q_k$.

As a fair girl, Alice hopes to divide fairly such that the lexicographical order of C is as small as possible. Please tell Alice the minimum possible C.

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 ($1 \leq n \leq 5 \times 10^3$) indicating the length of the sequence A.

The second line contains n integers $a_1, a_2, \cdots, a_n$ ($1 \leq a_i \leq 10^5$), where a_i is the i-th element of sequence A.

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

Output

For each test case output two lines. First output one line containing one integer m indicating the length of the optimal C. Then output a second line containing m integers $c_1, c_2, \cdots, c_m$ separated by a space, where c_i is the i-th element of the optimal C.

Example

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

Problem E. Elevator II

There is a building with 10^9 floors but only 1 elevator. Initially, the elevator is on the f-th floor.

There are n people waiting for the elevator. The i-th person is currently on the l_i-th floor and wants to take the elevator to the r_i-th floor ($l_i < r_i$). Because the elevator is so small, it can carry at most 1 person at a time.

It costs 1 unit of electric energy to move the elevator 1 floor upwards. No energy is needed if the elevator moves downwards. That is to say, it costs $\max(y - x, 0)$ units of electric energy to move the elevator from the x-th floor to the y-th floor.

Find the optimal order to take all people to their destinations so that the total electric energy cost is minimized.

More formally, let $a_1, a_2, \cdots, a_n$ be a permutation of n where a_i indicates that the i-th person to take the elevator is a_i. The total electric energy cost can be calculated as

$$\sum_{i=1}^{n}(\max(l_{a_i} - r_{a_{(i-1)}}, 0) + r_{a_i} - l_{a_i})$$

where $a_0 = 0, r_0 = f$ for convenience.

Recall that a sequence $a_1, a_2, \cdots, a_n$ of length n is a permutation of n if and only if each integer from 1 to n (both inclusive) appears exactly once in the sequence.

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 f ($1 \le n \le 10^5$, $1 \le f \le 10^9$) indicating the number of people and the initial position of the elevator.

For the following n lines, the i-th line contains two integers l_i and r_i ($1 \le l_i < r_i \le 10^9$) indicating that the i-th person wants to go from the l_i-th floor to the r_i-th floor by elevator.

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

Output

For each test case, first output one line containing one integer indicating the minimum total electric energy, then output another line containing n integers $a_1, a_2, \cdots, a_n$ separated by a space indicating the optimal order to carry all people. Note that these n integers must form a permutation of n. If there are multiple optimal orders, you can print any of them.

Example

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

Problem F. Fuzzy Ranking

In Pigeland, there are n universities, numbered from 1 to n. Each year, several ranking organizations publish rankings of these universities. This year, there are k ranking lists, where each list is a permutation of integers from 1 to n, representing the universities. In each ranking, the closer a university is to the beginning of the permutation, the better its ranking is in this list.

QS	Zhejiang University (#42)	>	Washington University in St. Louis (#118)
ARWU	Washington University in St. Louis (#23)	>	University of Michigan - Ann Arbor (#26)
QS	University of Michigan - Ann Arbor (#25)	>	University of Toronto (#34)
Times	University of Toronto (#18)	>	Cornell University (#22)
ARWU	Cornell University (#12)	>	University of Pennsylvania (#15)
QS	University of Pennsylvania (#13)	>	Princeton University (#16)
ARWU	Princeton University (#6)	>	California Institute of Technology - Caltech (#9)
Times	California Institute of Technology - Caltech (#2)	>	Massachusetts Institute of Technology - MIT (#5)
Therefore, Zhejiang University is better than Massachusetts Institute of Technology - MIT.			

A true story in the 2024 ICPC World Final.

Supigar, a year-4 student who wants to apply for PhD programs in Pigeland, has his own method to evaluate the n universities comprehensively. He considers that university x is *superior* to another university y if and only if:

- x is ranked better than y in at least one list, or
- x is ranked better than z ($z \neq x, z \neq y$) in at least one list, and z is superior to y.

Clearly, under this definition, there might exist some pairs of universities x and y ($x < y$) such that x is superior to y while y is also superior to x. Supigar calls such pairs *fuzzy*.

Supigar has q queries, where the i-th query can be represented by three integers id_i, l_i and r_i ($l_i \leq r_i$). For each query, he will consider the id_i-th rank list and all the universities between the l_i-th position and the r_i-th position (both inclusive) in that list. He wants to know, among these universities, how many pairs of them are fuzzy. Note that the definition of fuzzy pairs requires considering the superior relationships among all k rank lists.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \leq T \leq 2 \times 10^5$) indicating the number of test cases. For each test case:

The first line contains three integers n, k, and q ($1 \leq n, k, q \leq 2 \times 10^5$, $1 \leq n \times k \leq 2 \times 10^5$) indicating the number of universities, rank lists, and queries, respectively.

For the following k lines, the i-th line contains n distinct integers $a_{i,1}, a_{i,2}, \ldots, a_{i,n}$ ($1 \leq a_{i,j} \leq n$) indicating the i-th rank list.

For the following q lines, the i-th line contains three integers id'_i, l'_i, and r'_i ($0 \leq id'_i < k$, $0 \leq l'_i, r'_i < n$) indicating the **encoded** index of the rank list and the query range for the i-th query.

- The real value of id_i is equal to $((id'_i + v_{i-1}) \bmod k) + 1$.
- The real value of l_i is equal to $((l'_i + v_{i-1}) \bmod n) + 1$.
- The real value of r_i is equal to $((r'_i + v_{i-1}) \bmod n) + 1$.

Where v_{i-1} is the answer for the $(i-1)$-th query. Specifically, we define $v_0 = 0$. With the encoded queries, you're forced to calculate the answer to each query before processing the next one. It's guaranteed that $1 \le id_i \le k$ and $1 \le l_i \le r_i \le n$ after decoding.

It is also guaranteed that neither the sum of $n \times k$ nor the sum of q of all test cases will exceed 2×10^5.

Output

For each test case output q lines. Each line contains a single integer representing the number of fuzzy pairs as the answer to the i-th query.

Example

standard input	standard output
2 5 2 2 1 2 3 4 5 5 4 3 2 1 1 0 2 1 2 1 5 3 3 1 2 3 4 5 1 3 2 4 5 1 2 3 5 4 0 0 2 0 2 3 1 0 3	3 10 1 1 2

Note

For the first sample test case, the two decoded queries are `2 1 3` and `1 1 5`.

For the second sample test case, the three decoded queries are `1 1 3`, `2 4 5`, and `3 2 5`.

Problem G. Gathering Mushrooms

BaoBao is picking mushrooms in a forest. There are n locations in the forest, and in the i-th location there grows an infinite amount of mushrooms of type t_i. Each location also has a wooden sign. The sign of the i-th location points to location a_i (it is possible that $a_i = i$).

As it is very foggy in the forest, BaoBao decides to move between locations according to the signs just for safety. Starting from location s with an empty basket, each time BaoBao walks into a location c (including the starting location $c = s$, and regardless of whether he has visited location c before), he will pick one mushroom of type t_c into his basket and move to location a_c.

Given an integer k, for each $1 \le s \le n$, determine the first type of mushroom that appears at least k times in the basket.

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 k ($1 \le n \le 2 \times 10^5$, $1 \le k \le 10^9$) indicating the number of locations and the required times of appearance of mushrooms.

The second line contains n integers $t_1, t_2, \cdots, t_n$ ($1 \le t_i \le n$), where t_i is the type of mushroom growing in location i.

The third line contains n integers $a_1, a_2, \cdots, a_n$ ($1 \le a_i \le n$), where a_i is the location pointed to by the sign in location i.

It's guaranteed that the sum of n of all test cases will not exceed 2×10^5.

Output

To decrease the size of output, for each test case, output one line containing one integer indicating $\sum_{i=1}^{n}(i \times v_i)$, where v_i is the answer for $s = i$.

Example

standard input	standard output
3 5 3 2 2 1 3 3 2 5 1 2 4 5 4 2 2 1 3 3 2 5 1 2 4 3 10 1 2 3 1 3 2	41 45 14

Note

For the first sample test case, $v_1 = 2$, $v_2 = 3$, $v_3 = 2$, $v_4 = 3$, $v_5 = 3$, so you should output $1 \times 2 + 2 \times 3 + 3 \times 2 + 4 \times 3 + 5 \times 3 = 41$. Consider $s = 3$, the types of mushrooms BaoBao picks in order are $\{1, 2, 2, 3, 3, 2, \cdots\}$, so mushrooms of type 2 is the very first type which appears at least 3 times in the basket.

Problem H. Heavy-light Decomposition

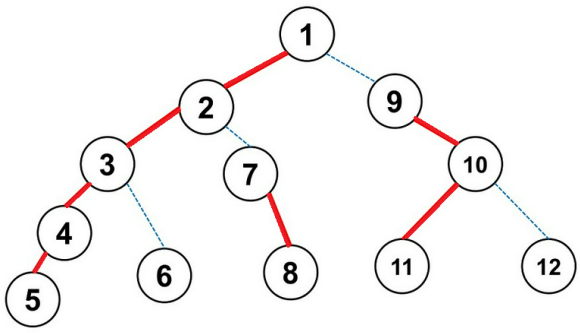
Heavy-light Decomposition (HLD) is a useful technique applied to trees for efficiently querying chains of vertices. Let's first review the definition of HLD in case you forget.

You are given a rooted tree with n vertices, numbered from 1 to n. You need to classify each non-root vertex as either heavy or light. Each non-leaf vertex has exactly one child classified as heavy, and the remaining vertices as light.

For any non-root vertex v, let u be its parent. Vertex v can be classified as heavy only if the size of its subtree is among the largest of all the children of u. More formally, denote the size of the subtree rooted at vertex v as s_v, and let $\mathrm{ch}(u)$ represent the set of children of u. Then, v can be heavy only if $s_v \geq s_w$ for all $w \in \mathrm{ch}(u)$. Note that there might be several children of u satisfying this constraint; in this case, you should choose one of them to be heavy, and the others to be light.

After that, all vertices of the tree can be decomposed into several non-overlapping heavy chains, where each vertex belongs to exactly one heavy chain. A heavy chain is a sequence of vertices $x_1, x_2, \ldots, x_k$ satisfying all the following constraints.

- x_1 is either the root or a light vertex.
- For all $2 \leq i \leq k$, x_i is x_{i-1}'s child and is a heavy vertex.
- x_k is a leaf.



An HLD example. Heavy chains are marked with solid red edges.

For example, the above figure shows a valid HLD of the given tree with 12 vertices, where the heavy chains are $[1, 2, 3, 4, 5]$, $[9, 10, 11]$, $[7, 8]$, $[6]$, and $[12]$.

Pig100Ton is a very experienced competitive programming contestant in Pigeland. He wants to know whether it is possible to recover the original tree from the heavy chains after HLD. Specifically, he will give you the number of vertices n of the original tree and k heavy chains. The i-th chain is described by two integers l_i and r_i, indicating a heavy chain $l_i, l_i + 1, \ldots, r_i$.

Your task is to construct a tree satisfying the following constraints or to tell Pig100Ton it is impossible:

- It is a rooted tree that contains n vertices numbered from 1 to n.
- The k chains provided by Pig100Ton should form a valid HLD of the tree. A valid HLD refers to a valid way to classify each non-root vertex as either light or heavy, and then decompose the tree into several non-overlapping heavy chains.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \leq T \leq 10^5$) indicating the number of test cases. For each test case:

The first line contains two integers n and k ($1 \le n \le 10^5$, $1 \le k \le n$), indicating the number of vertices of the tree and the number of heavy chains after its HLD.

For the following k lines, the i-th line contains two integers l_i and r_i ($1 \le l_i \le r_i \le n$), indicating that the i-th heavy chain is $l_i, l_i + 1, \ldots, r_i$.

It is guaranteed that each of the n vertices belongs to exactly one given heavy chain. It is also guaranteed that the sum of n of all test cases does not exceed 2×10^5.

Output

For each test case:

If it is possible to construct such a tree, output one line containing n integers $p_1, p_2, \cdots, p_n$ separated by a space, where p_i is the parent of vertex i. $p_i = 0$ indicates that vertex i is the root. If there are multiple valid solutions, you may output any one of them.

If it is not possible to construct such a tree, just output `IMPOSSIBLE` in one line.

Example

standard input	standard output
3 12 5 1 5 9 11 7 8 6 6 12 12 4 3 1 1 4 4 2 3 2 2 1 1 2 2	0 1 2 3 4 3 2 7 1 9 10 10 2 0 2 2 IMPOSSIBLE

Note

For the first test case, the sample output is the tree shown in the description.

Problem I. Identify Chord

This is an interactive problem.

Grammy has an undirected cyclic graph of n ($4 \le n \le 10^9$) vertices numbered from 1 to n. An undirected cyclic graph is a graph of n vertices and n undirected edges that form one cycle. Specifically, there is a bidirectional edge between vertex i and vertex $((i \bmod n) + 1)$ for each $1 \le i \le n$.

Grammy thinks that this graph is too boring, so she secretly chooses a pair of *non-adjacent* vertices and connects an undirected edge (called a chord) between them, so that the graph now contains n vertices and $(n + 1)$ edges.

Your task is to guess the position of the chord by making no more than 40 queries. Each query consists of two vertices x and y, and Grammy will tell you the number of edges on the shortest path between the two vertices.

Note that the interactor is *non-adaptive*, meaning that the position of the chord is pre-determined.

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 ($4 \le n \le 10^9$) indicating the number of vertices.

Interaction Protocol

To ask a query, output one line. First output `?` followed by a space, then output two vertices x and y ($1 \le x, y \le n$) separated by a space. After flushing your output, your program should read a single integer indicating the number of edges on the shortest path between the two vertices.

To guess the position of the chord, output one line. First output `!` followed by a space, then output two vertices u and v ($1 \le u, v \le n$) separated by a space, indicating that the chord connects vertices u and v. After flushing your output, your program should read a single integer r ($r \in \{1, -1\}$) indicating the correctness of your guess. If $r = 1$ then your guess is correct, and your program should continue processing the next test case, or exit immediately if there are no more test cases. Otherwise if $r = -1$ then your guess is incorrect, and your program should exit immediately to receive a `Wrong Answer` verdict. Note that your guess does not count as a query.

To flush your output, you may use:

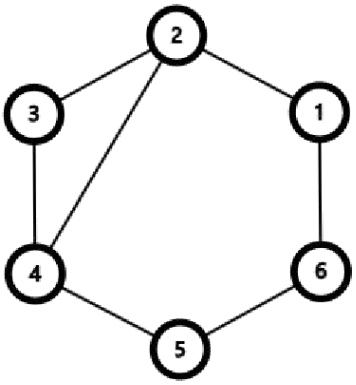
- `fflush(stdout)` (if you use `printf`) or `cout.flush()` (if you use `cout`) in C and C++.
- `System.out.flush()` in Java.
- `stdout.flush()` in Python.

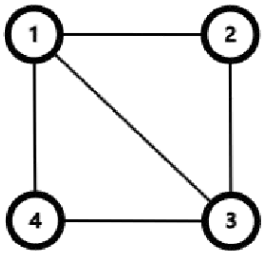
Example

standard input	standard output
2 6 2 1 1 4 2 1	 ? 1 5 ? 2 4 ! 4 2 ? 2 4 ! 1 3

Note

The graphs in the sample test cases are illustrated as follows:





Problem J. Japanese Bands

Grammy is designing a new trading card game (TCG) based on her favorite Japanese music media franchise, *BanG Dream!*. By her design, there are n_1 character cards and n_2 music cards in total. Now she needs to assign an integer between 1 and m (both inclusive) for each card, representing the magic power it contains.

In every TCG game, there must be some certain combos that may cause extra damage. Grammy now is considering a special rule that relates to the value assigned to cards. Specifically, k pairs of integers $(a_1, b_1), (a_2, b_2), \cdots, (a_k, b_k)$ are chosen, satisfying $1 \le a_i, b_i \le m$. Grammy wants to make sure each of these value combos can be played in her game. Therefore, for each pair of integers (a_i, b_i), the assignment must meet at least one of the two following constraints:

- a_i can be found on a character card and b_i can be found on a music card.
- a_i can be found on a music card and b_i can be found on a character card.

Please help Grammy count the number of valid card-value assignments.

Let $\mathbb{C}$ be the multi-set of the integers on the character card and $\mathbb{M}$ be the multi-set of the integers on the music card. We say two assignments are different if their $\mathbb{C}$s are different or their $\mathbb{M}$s are different.

Recall that an integer can appear multiple times in a multi-set. We say two multi-sets $\mathbb{X}$ and $\mathbb{Y}$ are different if there exists an integer k such that the number of times k appears in $\mathbb{X}$ is not equal to that in $\mathbb{Y}$.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 500$) indicating the number of test cases. For each test case:

The first line contains four integers n_1, n_2, m and k ($1 \le n_1, n_2 \le 10^9$, $1 \le m \le 20$, $1 \le k \le m^2$).

For the following k lines, the i-th line contains two integers a_i and b_i ($1 \le a_i, b_i \le m$).

It's guaranteed that there are at most 5 test cases satisfying $m > 10$.

Output

For each test case output one line containing one integer indicating the number of valid card-value assignments. As the answer may be large, output the answer modulo $(10^9 + 7)$.

Example

standard input	standard output
3 2 3 3 3 2 3 1 1 2 3 2 2 2 1 1 1 1 1 10 2 1 2 1 3	6 4 0

Note

For the first sample test case, the valid pairs of $(\mathbb{C}, \mathbb{M})$ are $(\{1,2\},\{1,1,3\})$, $(\{1,2\},\{1,2,3\})$, $(\{1,2\},\{1,3,3\})$, $(\{1,3\},\{1,1,2\})$, $(\{1,3\},\{1,2,2\})$ and $(\{1,3\},\{1,2,3\})$.

For the second sample test case, the valid pairs of $(\mathbb{C}, \mathbb{M})$ are $(\{1,1\},\{1,1\})$, $(\{1,2\},\{1,1\})$, $(\{1,1\},\{1,2\})$ and $(\{1,2\},\{1,2\})$.

Problem K. Kind of Bingo

There is a grid with n rows and m columns. The cells in the grid are numbered from 1 to $n \times m$, where the cell on the i-th row and the j-th column is numbered as $((i-1) \times m + j)$.

Given a permutation $p_1, p_2, \cdots, p_{n \times m}$ of $n \times m$, we're going to perform $n \times m$ operations according to the permutation. For the i-th operation, we'll mark cell p_i. If after the b-th operation, there is at least one row such that all the cells in that row are marked, and b is as small as possible, then we say b is the "bingo integer" of the permutation.

You're given the chance to modify the permutation at most k times (including zero times). Each time you can swap a pair of elements in the permutation. Calculate the smallest possible bingo integer after the modifications.

Recall that a sequence $p_1, p_2, \cdots, p_{n \times m}$ of length $n \times m$ is a permutation of $n \times m$ if and only if each integer from 1 to $n \times m$ (both inclusive) appears exactly once in the sequence.

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 ($1 \le n, m \le 10^5$, $1 \le n \times m \le 10^5$, $0 \le k \le 10^9$), indicating the number of rows and columns of the grid and the number of modifications you can perform.

The second line contains $n \times m$ distinct integers $p_1, p_2, \cdots, p_{n \times m}$ ($1 \le p_i \le n \times m$).

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

Output

For each test case output one line containing one integer indicating the smallest possible bingo integer after the modifications.

Example

standard input	standard output
3 3 5 2 1 4 13 6 8 11 14 2 7 10 3 15 9 5 12 2 3 0 1 6 4 3 5 2 2 3 1000000000 1 2 3 4 5 6	7 5 3

Note

For the first sample test case, we can first swap 1 and 15, then swap 6 and 12 to get the sequence $[15, 4, 13, 12, 8, 11, 14, 2, 7, 10, 3, 1, 9, 5, 6]$. It's easy to see that after the 7-th operation, all cells in the 3-rd row will be marked.

For the second sample test case, it's easy to see that after the 5-th operation, all cells in the 2-nd row will be marked.

For the third sample test case, we don't need to make any modifications. It's easy to see that after the 3-rd operation, all cells in the 1-st row will be marked.

Problem L. Let's Go! New Adventure

In Pigeland, *Pishin* is a popular open-world action RPG where users can play multiple characters. Each character has an independent *adventure rank*, which increases as they earn experience points (EXP) while being played. Initially, every character starts with an adventure rank of level 0 and can progress up to a maximum level of m. To advance from level $(i-1)$ to level i $(1 \le i \le m)$, the character is required to earn b_i EXP. The higher the current rank, the more difficult it becomes to level up, meaning $b_i \le b_{i+1}$ always holds for all i from 1 to m.

Grammy plans to play *Pishin* for the next n days. As a rich girl, her *Pishin* account has an infinite number of characters. However, being a lazy girl, all characters in her account start with an adventure rank of level 0 at the beginning of the n days. Each day, Grammy will select exactly one character to play, but once she stops playing a character, she cannot resume playing that character on any future day. In other words, she can only continue playing the same character on consecutive days.

On the i-th day, Grammy will earn a_i EXP for the character she plays. This means that if she plays a character continuously from the l-th day to the r-th day (both inclusive), the character's adventure rank will increase to level k, where k is the largest integer between 0 and m such that the total EXP earned (which is $\sum_{i=l}^{r} a_i$) is greater than or equal to the requirement of leveling up to k (which is $\sum_{i=1}^{k} b_i$).

Being a greedy girl, Grammy wants to maximize the total sum of adventure ranks across all her characters after the n days. However, as a single-minded girl, she doesn't want to play too many different characters. To balance this, she introduces a penalty factor of c. Her goal is to maximize the total sum of adventure ranks across all characters after the n days, minus $c \times d$, where d is the number of different characters she plays. As Grammy's best friend, your task is to compute the maximum value she can achieve under the optimal strategy for selecting characters.

Input

There are multiple test cases. The first line of the input contains an integer T $(1 \le T \le 5 \times 10^4)$ indicating the number of test cases. For each test case:

The first line contains three integers n, m and c $(1 \le n, m \le 5 \times 10^5,\ 0 \le c \le 5 \times 10^5)$.

The second line contains n integers $a_1, a_2, \cdots, a_n$ $(0 \le a_i \le 10^{12},\ 0 \le \sum_{i=1}^{n} a_i \le 10^{12})$.

The third line contains m integers $b_1, b_2, \cdots, b_m$ $(0 \le b_i \le 10^{12},\ b_i \le b_{i+1},\ 0 \le \sum_{i=1}^{m} b_i \le 10^{12})$.

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

Output

For each test case, output one line containing one integer, indicating the maximum value.

Example

standard input	standard output
2 5 4 2 1 0 3 1 2 0 1 1 2 4 5 1 7 16 23 4 1 3 6 20 20	3 6

Note

For the first sample test case, one solution is to use the first three days to get a character with adventure

rank 4 and the next two days to get another character with adventure rank 3. This gives us a value of $(4-2)+(3-2)=3$.

For the second sample test case, we can play a different character each day; this gives us adventure ranks 2, 3, 3, and 2, respectively. So the value is $(2-1)+(3-1)+(3-1)+(2-1)=6$.

Problem M. Make It Divisible

Given a sequence $a_1, a_2, \cdots, a_n$ of length n containing positive integers, we say an interval $[l, r]$ $(1 \le l \le r \le n)$ is a *divisible interval* if there exists an integer d such that $l \le d \le r$ and for all $l \le i \le r$, a_i is divisible by a_d. We say the whole sequence is a *divisible sequence* if for all $1 \le l \le r \le n$, $[l, r]$ is a divisible interval.

Given another sequence $b_1, b_2, \cdots, b_n$ of length n and an integer k, find all integers x such that $1 \le x \le k$ and the sequence $b_1 + x, b_2 + x, \cdots, b_n + x$ is a divisible sequence. As the number of such integers might be large, you just need to output the number and the sum of all such integers.

Input

There are multiple test cases. The first line of the input contains an integer T $(1 \le T \le 500)$ indicating the number of test cases. For each test case:

The first line contains two integers n and k $(1 \le n \le 5 \times 10^4,\ 1 \le k \le 10^9)$.

The second line contains n integers $b_1, b_2, \cdots, b_n$ $(1 \le b_i \le 10^9)$.

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

Output

For each test case output one line containing two integers separated by a space, where the first integer is the number of valid x, and the second integer is the sum of all valid x.

Example

standard input	standard output
3 5 10 7 79 1 7 1 2 1000000000 1 2 1 100 1000000000	3 8 0 0 100 5050

Note

For the first sample test case, $x = 1$, $x = 2$ and $x = 5$ are valid.

For the third sample test case, all $1 \le x \le 100$ are valid.