The 2nd Universal Cup

Stage Ω: Atlantis

$\mathcal{PRIME}$ Contest

December 24, 2023 - Jan 6, 2024

300 hours

This problem set should contain 39 problems on 68 numbered pages.

Tasks List

Here is the list of the tasks of the contest. It includes the source of each task.

A heartfelt thank you to the dedicated authors who meticulously prepared and presented the contests. Your contributions are the cornerstone of our event's success.

We also extend our deepest gratitude to all participants and supporters of the Universal Cup. Your unwavering support and enthusiasm has brought us this far.

As we celebrate this season of joy, we wish everyone a Merry Christmas and a Happy New Year. Happy Holidays, and enjoy the contest!

Task List

ID	Task	Source
2	Most Annoying Constructive Problem	OCPC Winter 2023. Anton Trygub Contest 1
3	Balanced Permutations	OCPC Winter 2023. Dilhan Salgado Contest
5	Mirror	The 2018 ICPC Qingdao Regional Contest
7	Best Problem	OCPC Winter 2023. Yuhao Du Contest 11
11	Cryptography Problem	OCPC Winter 2023. Yuhao Du Contest 11
13	Elliptic Curve Problem	OCPC Winter 2023. Yuhao Du Contest 11
17	Graph Problem	OCPC Winter 2023. Yuhao Du Contest 11
19	Junk Problem	OCPC Winter 2023. Yuhao Du Contest 11
23	The Best Problem of 2021	Petrozavodsk Winter 2023. Um_nik mod 998244353 Contest
29	LaLa and Magic Circle (LaLa Version)	OCPC Winter 2023. Magical Adventure of LaLa
31	Master of Polygon	The 2023 Zhejiang Collegiate Programming Contest
37	Are you a bot?	IOI 2023 China Mutual Test, by Siyuan Luo
41	Neighbourhood	The 2023 ICPC China Shaanxi Provincial Programming Contest
43	Grid Points	The 2022 ICPC Asia Jinan Regional Contest
47	Levenshtein Distance	The 2022 ICPC Asia Hangzhou Regional Contest
53	Square Game	2023 Nowcoder Multi University Training Contest 4
59	Be Careful	Petrozavodsk Summer 2022. Qingyu, flower, and their friends Contest
61	DS Team Selection	Petrozavodsk Summer 2022. Qingyu, flower, and their friends Contest
67	Half Plane	Petrozavodsk Summer 2022. Qingyu, flower, and their friends Contest
71	Kitten's Computer	Petrozavodsk Summer 2022. Qingyu, flower, and their friends Contest
73	Long: WCWBTT	Petrozavodsk Summer 2022. Qingyu, flower, and their friends Contest
79	Matrix Counting	Petrozavodsk Summer 2022. Qingyu, flower, and their friends Contest
83	Puzzle: Patrick's Parabox	Petrozavodsk Summer 2022. Heltion Contest
89	Poker Game: Construction	Petrozavodsk Summer 2022. Heltion Contest
97	Symmetry: Closure	Petrozavodsk Summer 2022. Heltion Contest
101	Noodle	Petrozavodsk Summer 2022. ZJU Contest 2
103	Geometry	Petrozavodsk Summer 2022. ZJU Contest 2
107	DFS	Petrozavodsk Summer 2022. ZJU Contest 2
109	Vacation	The 2021 ICPC Asia-East Continent Final Contest
113	Vision Test	The 2021 ICPC Asia-East Continent Final Contest
127	Sequence to Sequence	The 2017 Zhejiang Collegiate Programming Contest
131	Keychain	Petrozavodsk Summer 2023. olmrgcsi And His Friends' Contest
137	Greedy Bipartite Matching	Petrozavodsk Winter 2023. LOUD Enough Contest 2
139	Endless Road	Petrozavodsk Winter 2023. LOUD Enough Contest 2
149	LCSLCSLCS	Petrozavodsk Winter 2023. LOUD Enough Contest 2
151	Colonization	Potyczki Algorytmiczne 2022, final
157	Redundant Towers	CCPC 2023 Guilin Site
163	Rook Detection	The 2023 ICPC Asia Shenyang Regoinal Contest
167	Distinct Game	The 2023 ICPC Southeastern Europe Regional Contest

Problem 2. Most Annoying Constructive Problem

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

The array $a_1, a_2, \ldots, a_m$ of integers is called **odd** if it has an odd number of inversions, and **even** otherwise. Recall that an inversion is a pair (i, j) with $1 \le i < j \le m$ such that $a_i > a_j$. For example, in the array $[2, 4, 1, 3]$, there are 3 inversions: $(1, 3), (2, 3), (2, 4)$ (since $a_1 > a_3, a_2 > a_3, a_2 > a_4$), so it is **odd**.

Given n, k, determine if there exists a permutation of integers from 1 to n, which has exactly k odd subarrays.

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

Input

The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. The description of the test cases follows.

The only line of each test case contains two integers n, k ($1 \le n \le 1000, 0 \le k \le \frac{n(n-1)}{2}$).

It's guaranteed that the sum of n^2 over all test cases doesn't exceed $4 \cdot 10^6$.

Output

For every test case, if there is no such permutation, output `NO`.

Otherwise, output `YES`. In the next line, output n integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$, all p_i are distinct) — the elements of your permutation.

Example

standard input	standard output
4 1 0 3 3 4 1 6 15	YES 1 YES 3 2 1 YES 1 3 4 2 NO

Note

In the first test case, the permutation is (1); all its subarrays are even.

In the second test case, the permutation is $(3, 2, 1)$. It has 3 odd subarrays: $[3, 2], [2, 1]$ with 1 inversion each, and $[3, 2, 1]$ with 3 inversions.

In the third test case, the permutation is $(1, 3, 4, 2)$. It has exactly 1 odd subarrays: $[4, 2]$ with 1 inversion.

It can be shown that no such permutation exists for the fourth test case.

Problem 3. Balanced Permutations

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	256 megabytes

Given a permutation p of size n, let a (contiguous) subarray of p be 'unstable' if the maximum value contained within the subarray is its first or last element. A permutation is considered 'balanced' if it has the minimum number of 'unstable' subarrays over all permutations of size n.

Given integers n, l, and k, report the l-th lexicographically-minimum 'balanced' permutation and the k-th lexicographically-maximum 'balanced' permutation of size n. If no such permutation exists output -1 instead.

Input

The only line of input contains three integers n, l, and k ($1 \le n \le 10^5$, $1 \le l, k \le 10^{18}$) — the length of the desired permutation and the indices of which lex-min and lex-max permutation should be provided.

Output

Output two lines. The first line should contain the l-th lexicographically-minimum 'balanced' permutation of size n — denoted p.

The second line should contain the k-th lexicographically-maximum 'balanced' permutation of size n — denoted q.

p and q should satisfy $1 \le p_i, q_i \le n$ for all $1 \le i \le n$.

If p or q does not exist (ie. there do not exist l or k 'balanced' permutations of size n) then report an answer of -1 instead.

Examples

standard input	standard output
3 1 2	1 3 2 1 3 2
4 9 13	3 1 4 2 -1

Problem 5. Mirror

Input file:	`standard input`
Output file:	`standard output`
Time limit:	15 seconds
Memory limit:	1024 megabytes

There is a non-transparent obstacle and a single-sided mirror in an infinite two-dimensional plane. The obstacle can be represented as a triangle and the mirror can be represented as a **directional** line segment pointing from $(x_{m,1}, y_{m,1})$ to $(x_{m,2}, y_{m,2})$, with the right side being reflective.

There are m stones at point (x_1, y_1) and DreamGrid would like to move all the stones to point (x_2, y_2). The following constraints must be satisfied:

- DreamGrid can only carry one stone each time.
- Once DreamGrid picks up a stone, he can only put it down at point (x_2, y_2).
- Let L be the path DreamGrid walked, then for each point p on L, DreamGrid should see all the stones directly or through the mirror.

Note that:

- If some part of the line vision is inside the obstacle (it's ok that the line vision passes a point or edge of the obstacle), it's considered, that DreamGrid cannot see the stone with this line of vision.
- If one of the two endpoints of the mirror is on the line of vision, it's considered, that DreamGrid can see the stone both in the mirror and through the mirror.
- The reflection process is governed by laws of physics — the angle of incidence is equal to the angle of reflection. The incident ray is in the same half-plane as the reflected ray, relative to the mirror.
- If the line of vision is parallel to the mirror, reflection doesn't take place, and the mirror isn't regarded as an obstacle.
- DreamGrid cannot move into the obstacle but can move on the edges or the vertices of the obstacle.
- DreamGrid cannot move through the mirror but can move on the mirror. Note that if DreamGrid is on the line segment of the mirror other than the two endpoints, he can only see the side he comes from, and cannot see through the mirror.

DreamGrid would like to know the shortest distance to move all stones from (x_1, y_1) to (x_2, y_2).

Input

There are multiple test cases. The first line of input is an integer T (about 100), indicates the number of test cases. For each test case:

The first line contains one integer m ($1 \le m \le 10^6$), indicating the number of stones.

The second line contains four integers x_1, y_1, x_2 and y_2, indicating the start point and the target point.

The third line contains four integers $x_{m,1}$, $y_{m,1}$, $x_{m,2}$ and $y_{m,2}$, indicating the coordinates of the mirror.

Each of the next 3 lines has two integers x_i and y_i, indicating the coordinates of the vertices of the obstacle.

All the coordinates will not exceed 100 in absolute value. Both the start point and target point are outside the obstacle and the mirror. The mirror and the obstacle have no points in common.

It is guaranteed that no three points are collinear.

Output

For each test case, output a real number indicating the shortest distance, or output -1 if DreamGrid cannot move all the stones under the constraints.

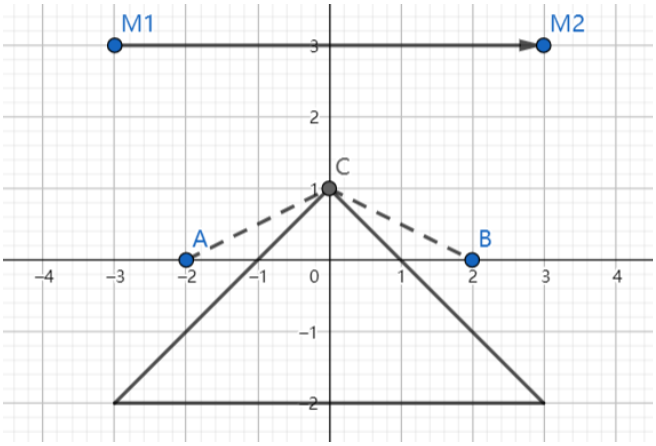
Your answer will be considered correct if and only if the absolute error or relative error of your answer is less than 10^{-6}.

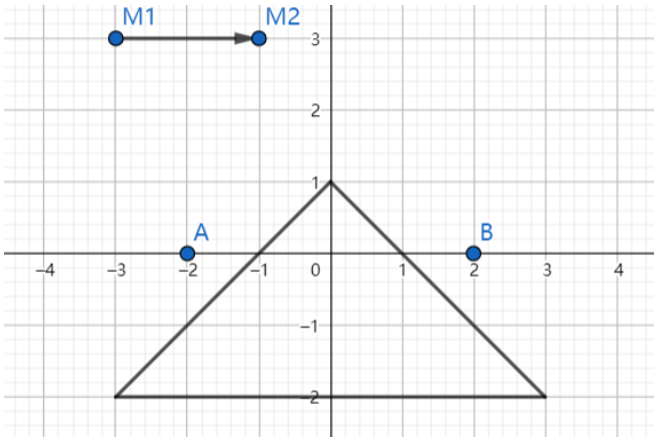
Example

standard input	standard output
2 2 -2 0 2 0 -3 3 3 3 0 1 -3 -2 3 -2 2 -2 0 2 0 -3 3 -1 3 0 1 -3 -2 3 -2	13.416407864999 -1

Note

We now welcome our special guest Mashiro, who heartily donates this problem to our problemset, to explain the sample test cases for us using her sketch book.





In the figures above, we indicate the start point as point A and the target point as point B. The mirror is indicated by the line segment pointing from $M1$ to $M2$, with the right side being reflective.

For the first sample test case, the optimal path is $A \to C \to B \to C \to A \to C \to B$.

For the second sample test case, as DreamGrid cannot see A from B, it's impossible to move all the two stones from A to B while following the constraints in the problem description.

(Image from pixiv. ID: 32084305)

Problem 7. Best Problem

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

You are given a binary string S. You can perform the following operation any number of times:

- Replace one substring '0101' with '1010'.

What is the maximum number of operations you can perform?

Input

A binary string S ($1 \le |S| \le 5 \times 10^6$).

Output

One integer — the answer.

Examples

standard input	standard output
10100010011001011111	5
000001010110011010110101010110000110100\ 111000010010111111100101101110101000 1\ 11101111010101010010101010 (There won't be extra line breakers \ in the actual test cases.)	58

Problem 11. Cryptography Problem

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	1024 megabytes

You are given m equations of the form

$$a_i \cdot x + err_i \equiv c_i \pmod{p}.$$

Here, err_i is an unknown random error term, chosen uniformly at random from $-\lfloor \frac{p}{200} \rfloor, \ldots, \lfloor \frac{p}{200} \rfloor$, while a_i, c_i and p are known to you.

You know that these equations hold for some unknown integer x. Find one such x.

Input

In the first line, T ($1 \le T \le 500$) — the number of test cases. For each test case:

- In the first line, m, p ($50 \le m \le 100, 10^{15} \le p \le 10^{18}$).
- In the next m lines, a_i, c_i ($0 \le a_i, c_i \le p - 1$).
- It's guaranteed that p is a prime, a_i, x are chosen uniformly at random from 0 to $p - 1$, and c_i is computed by $(a_i \cdot x + err_i) \bmod p$, err_i is an integer chosen uniformly at random from $-\lfloor \frac{p}{200} \rfloor, \ldots, \lfloor \frac{p}{200} \rfloor$.

Output

For each test case, one integer — the answer. If there are multiple solutions, you may output any.

Example

standard input	standard output
1 50 922033901407246477 492300877907148697 8585039545574817 36478175140515505 237143454432095134 537753813197233578 694568987600933631 ... (truncated)	578607642570710976

Note

The full sample test case is available in the contest system.

Problem 13. Elliptic Curve Problem

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

This problem might be well-known in some countries, but how do other countries learn about such problems if nobody poses them?

Let p be an odd prime. Compute the number of quadratic residues in $[l, r]$.

x is a quadratic residue of p iff $x^{(p-1)/2} \equiv 1 \pmod{p}$.

Input

In the first line, p, l, r ($3 \le p \le 10^{11}, 1 \le l \le r < p$). It's guaranteed that p is an odd prime.

Output

One integer — the answer.

Examples

standard input	standard output
11 3 8	3
998244353 11451400 919810000	454174074
96311898227 25437319919 55129361817	14846091352

Problem 17. Graph Problem

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	1024 megabytes

You are given a directed graph with n vertices and m edges. You want to answer q queries.

For each query, you are given $k_1, p_1, p_2, \ldots, p_{k_1}$, $k_2, s_1, t_1, s_2, t_2, \ldots, s_{k_2}, t_{k_2}$. For all i $(1 \le i \le k_2)$, answer whether there is a path from s_i to t_i if $p_1, p_2, \ldots, p_{k_1}$ are deleted. Queries are independent.

Input

In the first line, n, m $(1 \le n \le 500, 0 \le m \le n(n-1))$.

In the following m lines, u, v $(1 \le u, v \le n, u \ne v)$ — a directed edge in the graph. It's guaranteed that there is no parallel edges.

In the next line, q $(1 \le q \le 4 \times 10^5)$. To make sure you answer the queries online, the input is encrypted. The input can be decrypted using the following pseudocode:

```
cnt = 0
for i = 1 ... q
    read(k1)
    for j = 1 ... k1
        read(p'[j])
        p[j] =(p'[j] + cnt - 1) % n + 1
    read(k2)
    for j = 1 ... k2
        read(s', t')
        s = (s' + cnt - 1) % n + 1
        t = (t' + cnt - 1) % n + 1
        cnt += query(s, t)
// if s can reach t, query return 1, otherwise, query return 0
```

In the following $2q$ lines, for each query:

- In the first line, $k_1, p'_1, \ldots, p'_{k_1}$. It's guaranteed that p_i are distinct.
- In the second line, $k_2, s'_1, t'_1, \ldots, s'_{k_2}, t'_{k_2}$. It's guaranteed that all s_i, t_i are different from all p_i.
- It's guaranteed that $1 \le k_1 \le \min(n-2, 6), \sum k_2 \le 4 \times 10^6, 1 \le p'_i, s'_i, t'_i \le n$.

Output

For each query, output a binary string with length k_2 — the answer of query(s, t) in order.

Example

standard input	standard output
5 4 1 2 2 3 3 4 4 5 2 1 4 2 1 5 1 3 3 5 3 4 1 1 2	01 1

Note

It's recommended to use Fast IO.

Problem 19. Junk Problem

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

You are given a grid graph with n rows and m columns. Most edges are directed, which means you can walk from (x, y) to $(x+1, y)$ or $(x, y+1)$. k horizontal edges are bidirectional, which means you can walk from (x, y) to $(x, y+1)$, and $(x, y+1)$ to (x, y) too. It's guaranteed that there is no pair of bidirectional edges that share an endpoint.

You need to find l vertex-disjoint simple paths, where the i-th is from $(1, a_i)$ to (n, b_i). For a set of paths, we call a bidirectional edge *bad* if neither of its endpoints is visited by any of the paths in this set.

Output the number of all l vertex-disjoint simple paths without any bad edges, modulo 998244353.

Input

In the first line, n, m, l, k $(2 \le n, m \le 100, 1 \le l \le 50, 0 \le k \le 50)$.

In the second line, $a_1, a_2, \ldots, a_l$ $(1 \le a_1 < a_2 < \cdots < a_l \le m)$.

In the third line, $b_1, b_2, \ldots, b_l$ $(1 \le b_1 < b_2 < \cdots < b_l \le m)$.

In the following k lines, x_i, y_i $(1 \le x_i \le n, 1 \le y_i < m)$ each line, which denote that the edge (x_i, y_i) to $(x_i, y_i + 1)$ is bidirectional.

It's guaranteed that there is no pair of bidirectional edges that share an endpoint.

Output

One integer — the answer.

Examples

standard input	standard output
2 2 1 2 2 1 1 1 2 1	2
3 4 2 1 1 4 1 4 2 2	0
10 10 3 4 1 2 3 8 9 10 2 3 2 5 4 6 7 8	388035318

Problem 23. The Best Problem of 2021

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

The year is 2021. People still care about COVID, NNSU just won ICPC 2020, and he is already crazy, we just don't know yet how much. We are looking for the problems for the SnackDown finals and **7dan** suggested this one. For some reason, we decided not to use it then, but internally it became known as *The Best Problem of 2021*.

You are given an array B of numbers and a number X. Calculate (modulo 998 244 353, obviously) the number of subsets S of $\{1, 2, \ldots, X\}$ such that B is one of its bases if we consider the numbers to be vectors over $\mathbf{Z}_2$ with bitwise XOR as vector addition. B is considered to be a basis of S if it is an array of minimum size such that every element of S can be written as bitwise XOR of elements of B.

Input

The first line contains two integers n and m ($1 \le n, m \le 2000$) — the size of B and the length of our numbers in binary. All elements of B and the number X will be given in their binary representation with a length of exactly m (possibly with leading zeroes).

Each of the next n lines contains a binary string of length m which represents an element of B.

The last line contains a binary string of length m which represents the number X.

Output

You'll figure it out.

Examples

standard input	*standard output*
4 4 0001 0010 0100 1000 1101	7364
3 2 00 00 00 11	0
2 3 110 101 101	1
3 10 1111100110 0011110100 0101100001 1110000001	38

Problem 29. LaLa and Magic Circle (LaLa Version)

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

LaLa has a pile of magic circles in her laboratory.

A magic circle can be represented as a simple polygon drawn with special ink, and is **usable** if and only if it is convex. i.e. all of its internal angles are equal or less than π.

LaLa plans to turn every magic circle into a usable one. However, it may lose all its magical power if done incorrectly. Thankfully, LaLa has the perfect magical tool for that.

The tool works as follows. When you toss in a magic circle, if it's usable, it reports that it is. Otherwise, it takes two distinct points u and v such that

- u and v lie on the boundary of the convex hull of the magic circle, and
- none of the points on the path from u to v through the boundary of the magic circle in counterclockwise order lie on the boundary of the convex hull of the magic circle, except for u and v.

And then it rotates the u-v path by π around the midpoint of u and v. In other words, for each point w on the u-v path, w becomes $u + v - w$ where the addition is done coordinate-wise over the two dimensional coordinate system over the paper the magic circle is drawn on. Note that the result of this modification is also a simple polygon.

LaLa got annoyed by how long it takes to convert them. In order to finish and take a nap ASAP, LaLa made the following observations.

1. A magic circle always turns into a usable one within a finite number of applications of the tool.
2. The set of points on the final magic circle is independent of the intermediate modifications. In other words, the shape and location of the final magic circle is a function of the initial magic circle.

Therefore, LaLa doesn't have to manually turn magic circles into usable ones with the tool. Instead, LaLa will compute the usable magic circle that can be made from the initial magic circle by a sequence of modifications by the tool and modify it in one go.

Write a program to help LaLa compute the final magic circle so that she can go take a nap.

Input

The input is given in the following format:

$$\begin{array}{ll} N & \\ x_0 & y_0 \\ x_1 & y_1 \\ & \vdots \\ x_{N-1} & y_{N-1} \end{array}$$

where the initial magic circle is the union of N line segments connecting points (x_i, y_i) and $(x_{(i+1 \bmod N)}, y_{(i+1 \bmod N)})$ for all integers $0 \leq i < N$.

The input satisfies the following constraints:

- All numbers in the input are integers.

- $3 \le N \le 100\,000$
- $0 \le x_i \le 300\,000$ and $0 \le y_i \le 300\,000$ for all integers $0 \le i < N$.
- $x_i \ne x_j$ or $y_i \ne y_j$ for all integers $0 \le i < j < N$.
- The input defines a counterclockwise traversal of the boundary of a simple polygon. In particular, it does not intersect with itself.

Output

The output should be in the following format:

$$\begin{array}{ll} M & \\ z_0 & w_0 \\ z_1 & w_1 \\ & \vdots \\ z_{M-1} & w_{M-1} \end{array}$$

where the final usable magic circle is the union of M line segments connecting points (z_i, w_i) and $(z_{(i+1 \bmod M)}, w_{(i+1 \bmod M)})$ for all integers $0 \le i < M$,

The output should satisfy the following constraints:

- All the numbers in the output are integers.
- The point (z_0, w_0) is lexicographically smaller than the point (z_i, w_i) for all integers $1 \le i < M$. i.e. $(z_0 < z_i)$ or $(z_0 = z_i$ and $w_0 < w_i)$.
- Points (z_i, w_i), $(z_{(i+1 \bmod M)}, w_{(i+1 \bmod M)})$, and $(z_{(i+2 \bmod M)}, w_{(i+2 \bmod M)})$ are not collinear for all integers $0 \le i < M$.
- The output defines a counterclockwise traversal of the boundary of a convex polygon.

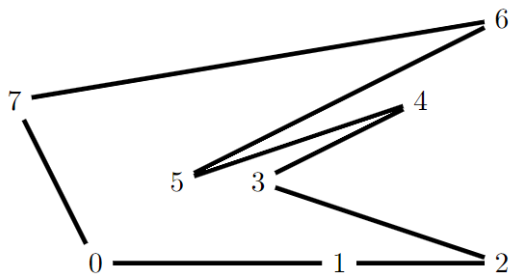
It can be proved that the output satisfying the above constraints is unique.

Example

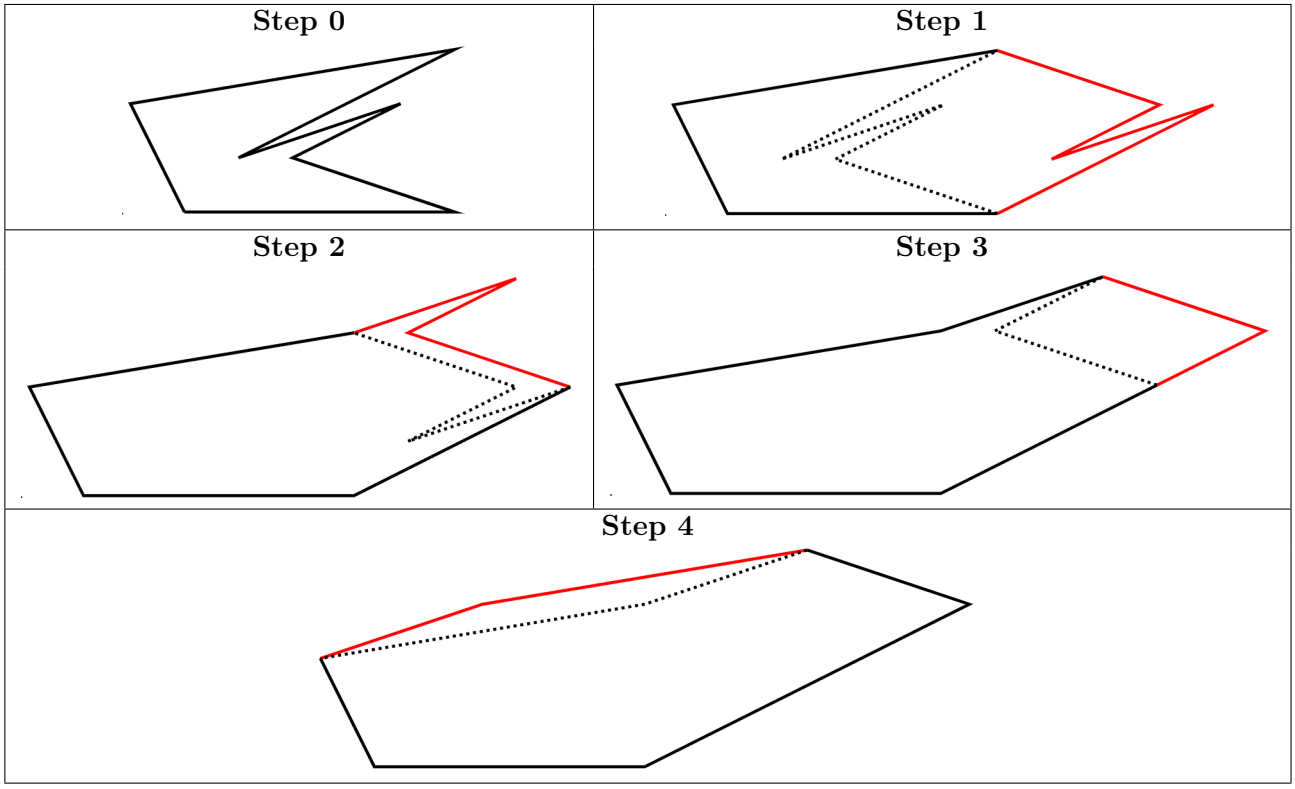
standard input	standard output
8 1 0 4 0 6 0 3 1 5 2 2 1 6 3 0 2	6 0 2 1 0 6 0 12 3 9 4 3 3

Note

The following illustrates the initial magic circle for the first sample.



The following illustrates the sequence of usage of the tool to make it usable. The dotted part of the boundary is the path modified by the tool, which becomes the red part after the modification.



Problem 31. Master of Polygon

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	1024 megabytes

Prof.Chen is the master of computational geometry. Now he has a simple polygon with n vertices lying on the Euclidean plane, he would like to give you q queries. In each query, you will be given two points P and Q, you need to check whether the segment PQ intersects with the boundary of the given polygon. Note that even when the segment touches a point of the polygon, you should also answer "`YES`".

Input

The first line of the input contains two integers n and q ($3 \le n \le 200\,000$, $1 \le q \le 200\,000$), denoting the number of vertices and the number of queries.

The next n lines, each line contains two integers x and y ($0 \le x, y \le 30\,000$) that give the coordinates (x, y) of the vertices of the polygon in either clockwise order or counter-clockwise order. The polygon is simple, i.e., its vertices are distinct and no two edges of the polygon intersect or touch, except that consecutive edges touch at their common vertex. In addition, no two consecutive edges are collinear.

Each of the next q lines contains four integers x_1, y_1, x_2 and y_2 ($0 \le x_1, y_1, x_2, y_2 \le 30\,000$), denoting a query segment with endpoint $P(x_1, y_1)$ and $Q(x_2, y_2)$. It's guaranteed that the two endpoints of each segment do not coincide.

Output

For each query, print "`YES`" or "`NO`" in a single line.

Example

standard input	standard output
4 6 1 1 4 1 4 4 1 4 0 2 2 0 0 1 1 1 0 0 5 5 2 2 4 2 2 2 3 2 5 1 0 2	YES YES YES YES NO YES

Problem 37. Are you a bot?

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

"What does the heartbeat of a bot, arranged into a graph, look like?"

You have a competitive programming bot, whose heart beats n times per minute. The intensity of the i-th heartbeat is a_i. Here, $a_1 \sim a_n$ is a permutation of $1 \sim n$.

Let A_i be the sequence obtained by deleting the i-th element from the sequence a, i.e., $A_i = [a_1, \cdots, a_{i-1}, a_{i+1}, \cdots, a_n]$.

For a sequence p of distinct elements, let $G(p)$ be an undirected graph with $|p|$ vertices, numbered $1 \sim |p|$. For every pair of positive integers $1 \le i < j \le |p|$, if $\forall k \in [i, j] \cap \mathbb{Z}$, we have $p_k \in [\min(p_i, p_j), \max(p_i, p_j)]$, then in $G(p)$, there is an edge between vertices i and j. Let $F(p)$ be the shortest path length from vertex 1 to vertex $|p|$ in $G(p)$, where a path length is defined as its number of edges.

Let $f(a) = [F(A_1), F(A_2), \ldots, F(A_n)]$.

Given a sequence of length n as $[b_1, \cdots, b_n]$, your task is to find any permutation a of $1 \sim n$ such that $f(a) = b$.

It is guaranteed that at least one solution exists.

Input

There are multiple test cases in a single test file.

The first line of the input contains a single integer T ($1 \le T \le 40\,000$), indicating the number of the test cases.

For each of the test case:

- The first line contains a single integer n ($4 \le n \le 10^5$).
- The next line contains n integers $b_1, b_2, \cdots, b_n$.
- It is guaranteed that at least one solution exists.

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

Output

For each test case, output a single line contains n integers $a_1, a_2, \cdots, a_n$, indicating the permutation you found.

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

Example

standard input	standard output
11 4 2 2 1 1 4 2 2 2 2 4 2 1 1 2 7 5 5 4 4 4 5 5 7 1 3 2 2 2 2 4 7 3 3 2 4 4 5 3 8 2 2 3 5 3 3 3 4 8 5 4 4 4 4 6 6 5 8 4 4 4 2 4 4 2 3 9 4 7 5 5 5 5 3 4 4 9 3 4 4 4 4 4 4 4 6	1 2 4 3 2 1 4 3 1 3 2 4 3 1 7 2 6 4 5 3 1 6 4 2 5 7 2 3 1 6 4 7 5 5 6 3 1 7 4 2 8 1 8 2 7 3 5 6 4 6 3 2 7 4 5 1 8 5 8 6 3 7 1 9 2 4 8 1 7 9 2 5 3 4 6

Problem 41. Neighbourhood

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	1024 megabytes

You are given a tree with n vertices. Each edge has a weight w_i.

There are q operations of the following two types:

- 1 i c: Change w_i to c.
- 2 x d: Count number of $y(1 \le y \le n)$ such that the shortest path between x and y is not greater than d.

Input

The first line of the input contains two integers n and q ($2 \le n \le 2 \times 10^5, 1 \le q \le 2 \times 10^5$).

The next $n-1$ lines, each line contains three integers x_i, y_i, w_i ($1 \le x_i, y_i \le n, 1 \le w_i \le 10^9$), representing an edge connecting x_i and y_i with weight w_i.

The next q lines, each line contains three integers 1 i c ($1 \le i \le n-1, 1 \le c \le 10^9$) or 2 x d ($1 \le x \le n, 0 \le d \le 2 \times 10^{14}$), indicating an operation.

Output

For each operation of type 2, print one line with a single integer, indicating the answer.

Example

standard input	standard output
3 7 1 2 3 2 3 1 2 2 1 2 1 3 2 3 4 1 1 1 2 2 1 2 1 0 2 3 1	2 2 3 3 1 2

Problem 43. Grid Points

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

You are given a simple polygon in the first quadrant of the Cartesian plane. This means that the coordinate (x, y) of any point in the polygon satisfies $x > 0$ and $y > 0$.

Consider all grid points in the polygon. Order them in increasing order of **slopes**. Output the k-th grid point in this order.

A grid point is a point (x, y) such that x and y are integers. A point on the boundary of a polygon is considered to be in the polygon. The slope of point (x, y) is y/x. If two points have the same slope, they are ordered lexicographically first by x, then by y. In other words, a point (x_1, y_1) is lexicographically less than (x_2, y_2) if $(x_1 < x_2) \vee (x_1 = x_2 \wedge y_1 < y_2)$.

Input

The first line contains one integer T ($1 \le T \le 500$), the number of test cases.

For each test case, the first line contains two integers n, k ($n \ge 3, 1 \le k$). Each of the next n lines describes a vertex of the polygon. Each vertex is denoted by two integers x, y ($1 \le x, y \le 10^9$), representing its coordinate (x, y). The vertices are given in counterclockwise order.

It is guaranteed that the polygon is simple, i.e. the vertices are distinct, two edges may overlap only when they are consecutive on the boundary, and the overlap contains exactly 1 point. It is guaranteed that k is no more than the number of grid points in the polygon.

It is guaranteed that the sum of n over all test cases is no more than 2000.

Output

For each test case, output two integers x, y in one line representing the coordinate (x, y) of the k-th grid point.

Example

standard input	standard output
4 3 3 1 1 3 1 3 3 4 500000000000000000 1 1 1000000000 1 1000000000 1000000000 1 1000000000 9 22 9 6 6 7 9 7 10 10 6 9 3 9 1 6 1 5 7 3 5 22447972861454999 270353376 593874603 230208698 598303091 237630296 255016434 782669452 568066304 654623868 958264153	3 2 500000000 500000000 7 8 730715389 644702744

Problem 47. Levenshtein Distance

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	1024 megabytes

The **Levenshtein Distance** between two strings is the smallest number of simple one-letter operations needed to change one string to the other. The operations are:

- Adding a letter anywhere in the string.
- Removing a letter from anywhere in the string.
- Changing any letter in the string to any other letter.

You will be given a number k and two strings S and T. Your task is to find the number of non-empty substrings of T whose Levenshtein Distance between S is exactly i for every possible non-negative integer i ($0 \le i \le k$). Two substrings are considered different if and only if they occur in different places.

Input

The first line contains a single integer k ($0 \le k \le 30$), denoting the parameter k.

The second line contains a string S ($1 \le |S| \le 10^5$), denoting the pattern string.

The third line contains a string T ($1 \le |T| \le 10^5$), denoting the text string.

It is guaranteed that the input strings only consist of lowercase English letters ('`a`' to '`z`'), uppercase English letters ('`A`' to '`Z`'), and digits ('`0`' to '`9`').

Output

Output $k+1$ lines, the i-th ($1 \le i \le k+1$) of which contains an integer denoting the number of substrings of T whose Levenshtein Distance between S is exactly $i-1$.

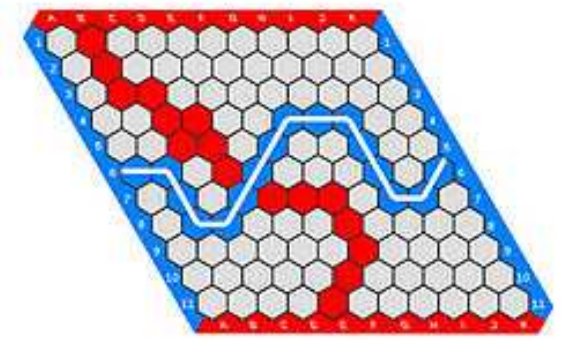
Example

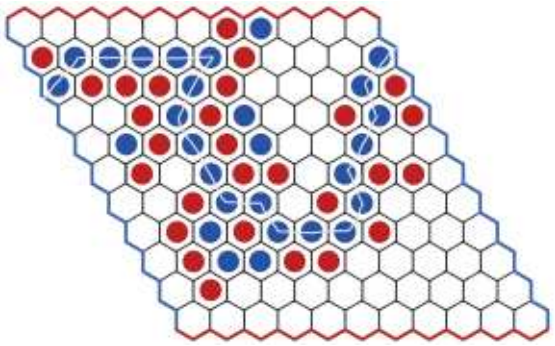
standard input	standard output
4 aaa aabbaab	0 5 15 7 1

Problem 53. Square Game

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

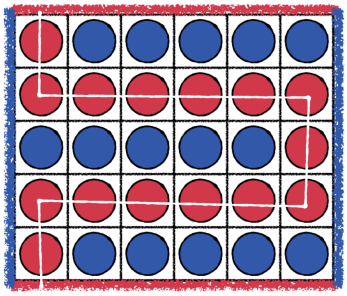
Have you ever heard of the game Hex? Hex is a two-player game played on a hexagonal board. The rules are that the red/blue players take turns placing their colored pieces on empty spaces on the board until the red player connects the top and bottom of the board or the blue player connects the left and right of the board. The first player to achieve their goal wins. Here the connection condition is that there must be a path of the player's color connecting two adjacent hexagons with at least one common edge. For example, the following two Hex game boards are both victories for the blue player.

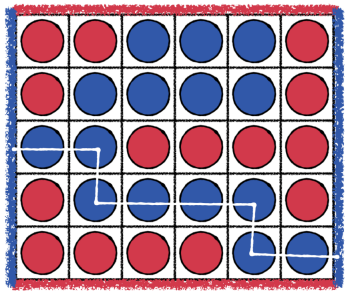




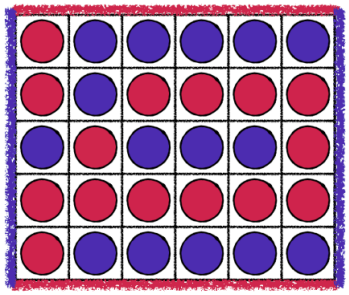
It can be proven that there is only one winning player in Hex.

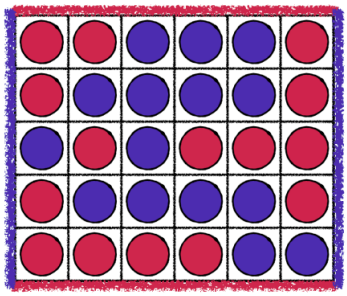
Bobo also wants to play Hex, but he thinks that having hexagonal cells on the board is too complicated. So, he invented a new game called Square! The rules of Square are similar to Hex, where the red/blue players take turns placing their colored pieces on empty spaces on the board until the red player connects the top and bottom of the board or the blue player connects the left and right of the board. However, in Square, the board is a rectangle made up of many small squares, and the connection condition remains the same, which is to have a path of the player's color connecting two adjacent squares with at least one common edge (i.e., four-connected). The following two Square boards on a 5×6 board are victories for the red and blue players, respectively.





Bobo was excited to share this Square game with his friend, oboB. However, oboB immediately realized a problem: Bobo's Square game may result in a tie! For example, the following two Square game boards on a 5×6 board are both ties because both players have no moves left, and the red pieces are not connected to the top and bottom of the board, and the blue pieces are not connected to the left and right of the board.





"But...there shouldn't be many tie games, right?" Bobo said stubbornly. To refute Bobo, oboB drew a n-row by m-column matrix of red and blue cells. "I randomly select a submatrix from here as the game board, and there is a high probability that it will be a tie!" Bobo seemed to agree, but he suddenly thought of something else, "In order to make the rules of Square valid, the submatrix selected must have all red cells at the top and bottom and all blue cells on the left and right! (corresponding to the red/blue lines in the example matrix)" oboB had no choice but to agree with Bobo's argument, "Then could you please count how many game boards satisfy this condition?"

Formally, given an n-row by m-column matrix A, where each cell is either red ('R') or blue ('B'), you need to count the number of quadruples (x_1, x_2, y_1, y_2) satisfying the following conditions (where (x_1, y_1) and (x_2, y_2) are the coordinates of the top-left and bottom-right corners of the submatrix you select):

1. $2 \le x_1 \le x_2 \le n-1$, $2 \le y_1 \le y_2 \le m-1$ (you must select a submatrix)
2. $\forall y_1 \le j \le y_2$, $A_{x_1-1,j} = A_{x_2+1,j} =$'R' (the top and bottom of the submatrix must be all red)
3. $\forall x_1 \le i \le x_2$, $A_{i,y_1-1} = A_{i,y_2+1} =$'B' (the left and right of the submatrix must be all blue)
4. There does not exist a sequence of cells $P = ((i_1, j_1), (i_2, j_2), \ldots, (i_\ell, j_\ell))$ satisfying:
   (a) $\ell \ge 1$, $i_1 = x_1$, $i_\ell = x_2$
   (b) $\forall 1 \le k \le \ell$, $x_1 \le i_k \le x_2 \wedge y_1 \le j_k \le y_2$
   (c) $\forall 1 \le k \le \ell$, $A_{i_k,j_k} =$'R'
   (d) $\forall 1 \le k \le \ell - 1$, $|i_k - i_{k+1}| + |j_k - j_{k+1}| = 1$

   (there is no connected submatrix with a red path between the top and the bottom)
5. There does not exist a sequence of cells $P = ((i_1, j_1), (i_2, j_2), \ldots, (i_\ell, j_\ell))$ satisfying:
   (a) $\ell \ge 1$, $j_1 = y_1$, $j_\ell = y_2$
   (b) $\forall 1 \le k \le \ell$, $x_1 \le i_k \le x_2 \wedge y_1 \le j_k \le y_2$
   (c) $\forall 1 \le k \le \ell$, $A_{i_k,j_k} =$'B'
   (d) $\forall 1 \le k \le \ell - 1$, $|i_k - i_{k+1}| + |j_k - j_{k+1}| = 1$

   (there is no connected submatrix with a blue path between the left and the right)

Input

The first line contains two integers n and m ($1 \le n, m \le 1500$), denoting the number of rows and columns of the matrix, respectively. In the following n lines, the ith ($1 \le i \le n$) line contains a string of length m, consisting only of 'R' and 'B'. The j-th ($1 \le j \le m$) character in the i-th line denotes the color of the cell in the i-th row and $j-$th column of the matrix, where 'R' stands for red and 'B' stands for blue.

Output

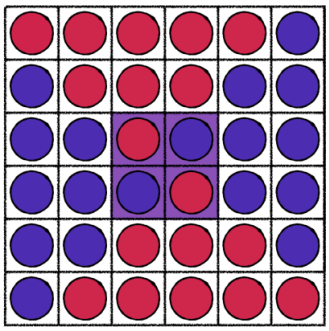
Output an integer in a line, denoting the number of submatrices that form a tie in a Square board (the formal definition has already been given in the problem statement).

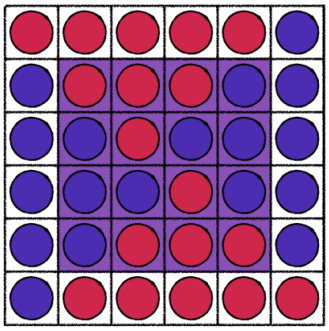
Examples

standard input	standard output
6 6 RRRRRB BRRRBB BBRBBB BBBRBB BBRRRB BRRRRR	2
5 5 BRRRB BBRRB BRBRB BRRBB BRRRB	1
7 7 BRRRRRB BRBBBRB BBRBRBB BRRRRRB BBRBRBB BRBBBRB BRRRRRB	7
3 3 RRR BRB RRR	0
20 20 RRRRRRRRRRRRRRRRRRRR BRRRRRRRRRRRRRRRRRRB BBBBBBBRRBBBBRRRRRRB BRRRRBBBBRRBRRRRRRRB BRRRRRRRRRRBRRRRRRRB BBBBBBBBBBBBRRRRRRRB BRRRRRRRRRRRRRRRRRRB BRRRRRRRRRRRRRRRRRRB BBBBBBBBBBRRRRRRRRRB BRRRRRRRBRBBBRRRRRB BRRRRRRRBBRRBRRRRRB BRRRRRRRRRRRRBRRRBBB BRRRRRRRRRRRRBRBBBRB BBBBBBBRRRRRRBBBRRRB BRRRRRBRRRRRRRRRRRRB BRRRRRBRRRRRRRRRRRRB BRRRRRBRRRRRRRRRRRRB BBBBBBBBBBRRRRRRRRRB BRRRRRRRRRRRRRRRRRRB RRRRRRRRRRRRRRRRRRRR	0

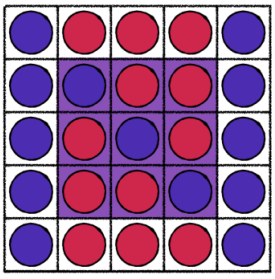
Note

To aid understanding, we provide graphical illustrations for the first two sample tests. The two valid submatrices in the first sample test are shown below.





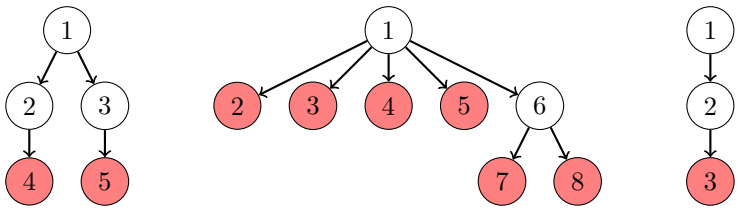
One valid submatrix in the second sample test is shown below.



Problem 59. Be Careful

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

You are given a rooted tree with n vertices, where the root is vertex 1. A vertex is a *leaf* if it is not the root vertex and its degree is exactly 1.



The figure corresponds to the sample tests, where the leaves are marked red.

Let $\operatorname{mex}(S)$ be the minimal non-negative integer that is not present in S. For example, $\operatorname{mex}\{0,1,3,4\} = 2$, $\operatorname{mex}\{2,3\} = 0$, $\operatorname{mex} \varnothing = 0$.

Let m be the number of leaves in the given tree. You will perform the following procedure:

1. For every **leaf vertex** u, write any integer from $\{0, 1, 2, \ldots, n\}$ to the vertex u.
2. For every **non-leaf vertex** u, the integer written in u will be the mex of the integers written in all the sons of vertex u.

For example, for the first tree which is described in the figure above, if we write integer 0 to vertex 4 and integer 3 to vertex 5, then:

- The integer written in vertex 2 will be $\operatorname{mex}\{0\} = 1$.
- The integer written in vertex 3 will be $\operatorname{mex}\{3\} = 0$.
- The integer written in vertex 1 will be $\operatorname{mex}\{1,0\} = 2$.

In total, there are $(n+1)^m$ ways to fill the tree. You would like to know, for all $k \in \{0, 1, 2, \ldots, n\}$, how many ways are there to fill the tree so that the number written in vertex 1 will be exactly k. Since the numbers can be huge, you only need to output them modulo 998 244 353.

Input

The first line of the input consists of a single integer n ($2 \le n \le 200$).

Each of the next $n-1$ lines contains two integers x and y ($1 \le x, y \le n$, $x \ne y$), indicating that there is an edge between vertices x and y. It is guaranteed that the given graph is a tree.

Output

Output $n+1$ lines. In the i-th line output a single integer, indicating the answer for $k = i - 1$, modulo 998 244 353.

Examples

standard input	*standard output*
5 1 2 1 3 2 4 2 5	55 127 34 0 0 0
8 1 2 1 3 1 4 1 5 1 6 6 7 6 8	69632 265534 133905 47790 12636 1944 0 0 0
3 1 2 2 3	1 3 0 0

Problem 61. DS Team Selection

Input file:	*standard input*
Output file:	*standard output*
Time limit:	18 seconds
Memory limit:	1024 mebibytes

The 34th *International Olympiad in Data Structures* will take place soon! In order to qualify, you need to pass the team selection contest in your country. As a member of the *Cat* team, you have to solve this problem in the *Cat* Team Selection contest.

There are infinitely many points **with integer coordinates** on an infinite plane, each of which can be represented as (x, y). Initially, the weights of all points are 0. You need to perform q operations, each of which takes the form:

- `1` x y d w: For all points (X, Y) that satisfy $|X - x| < d$ and $|Y - y| < d$, increase their point weights by $w \cdot (d - \max(|X - x|, |Y - y|))$.
- `2` x_1 x_2 y_1 y_2: Print the sum of the weights of points (x, y) that satisfy $x_1 \le x \le x_2$ and $y_1 \le y \le y_2$. Since the sum can be large, output it modulo 2^{30}.

Input

The first line contains a single integer m ($1 \le m \le 10^5$), indicating the number of the operations.

The next m lines contains several integers in one of the following forms:

- `1` x y d w ($1 \le x, y, d, w \le 10^8$)
- `2` x_1 x_2 y_1 y_2 ($1 \le x_1 \le x_2 \le 10^8$, $1 \le y_1 \le y_2 \le 10^8$)

Output

For each operation of type 2, print a single line containing an integer: the desired sum of the weights modulo 2^{30}.

Example

standard input	*standard output*
4 1 3 4 5 1 2 1 4 3 5 1 2 4 2 2 2 4 5 3 5	46 21

Problem 67. Half Plane

Input file:	*standard input*
Output file:	*standard output*
Time limit:	12 seconds
Memory limit:	1024 mebibytes

This problem might be well-known in some countries, but how do other countries learn about such problems if nobody poses them?

There are n points on the plane, where the i-th point (x_i, y_i) has value $\mathbf{d}_i \in D$. Two sets D and O are given, with the following properties:

- There exists a special element ε_D in D.
- There exists a special element ε_O in O.
- A binary operation $+ : D \times D \to D$ is given with the following properties:
  - $\forall \mathbf{a}, \mathbf{b} \in D$, $\mathbf{a} + \mathbf{b} = \mathbf{b} + \mathbf{a}$
  - $\forall \mathbf{a}, \mathbf{b}, \mathbf{c} \in D$, $(\mathbf{a} + \mathbf{b}) + \mathbf{c} = \mathbf{a} + (\mathbf{b} + \mathbf{c})$
  - $\forall \mathbf{x} \in D$, $\mathbf{x} + \varepsilon_D = \varepsilon_D + \mathbf{x} = \mathbf{x}$
- A binary operation $\cdot : O \times D \to D$ is given with the following properties:
  - $\forall \mathbf{a}, \mathbf{b} \in O, \mathbf{x} \in D$, $(\mathbf{a} \cdot \mathbf{b}) \cdot \mathbf{x} = \mathbf{a} \cdot (\mathbf{b} \cdot \mathbf{x})$
  - $\forall \mathbf{a} \in O, \mathbf{x}, \mathbf{y} \in D$, $\mathbf{a} \cdot (\mathbf{x} + \mathbf{y}) = \mathbf{a} \cdot \mathbf{x} + \mathbf{a} \cdot \mathbf{y}$
- A binary operation $\cdot : O \times O \to O$ is given with the following properties:
  - $\forall \mathbf{a}, \mathbf{b}, \mathbf{c} \in O$, $(\mathbf{a} \cdot \mathbf{b}) \cdot \mathbf{c} = \mathbf{a} \cdot (\mathbf{b} \cdot \mathbf{c})$
  - $\forall \mathbf{x} \in O$, $\mathbf{x} \cdot \varepsilon_O = \varepsilon_O \cdot \mathbf{x} = \mathbf{x}$

In this problem, we treat D as the set of all 3×1 matrices over $\mathbb{F}_p$ and O as the set of all 3×3 matrices over $\mathbb{F}_p$, where $p = 10^9 + 7$. That is, you can treat the above operations as the usual matrix addition and matrix multiplication modulo $10^9 + 7$.

Now, m queries are given in the form `a b c o`:

- Let $\mathbf{s} = \varepsilon_D$.
- For all points i with $ax_i + by_i < c$, modify $\mathbf{s}$ to $\mathbf{s} + \mathbf{d}_i$, then modify $\mathbf{d}_i$ to $\mathbf{o} \cdot \mathbf{d}_i$.
- Return $\mathbf{s}$ as the answer of the query.

As a data structure master, you need to perform all queries and find the answer.

Input

The first line of the input contains a single integer n ($1 \le n \le 3 \cdot 10^5$), indicating the number of points.

Each of the following n lines contains **five** integers $x_i, y_i, d_{i0}, d_{i1}, d_{i2}$, indicating the coordinates of the i-th point and its value $\mathbf{d}_i = \begin{bmatrix} d_{i0} \\ d_{i1} \\ d_{i2} \end{bmatrix}$.

The next line of the input contains a single integer m ($1 \le m \le 1.5 \cdot 10^4$), indicating the number of the queries.

Each of the following m lines contains **twelve** integers $a, b, c, o_{00}, o_{01}, o_{02}, o_{10}, \ldots, o_{22}$. Note that the real $\mathbf{o} = \begin{bmatrix} o_{00} & o_{01} & o_{02} \\ o_{10} & o_{11} & o_{12} \\ o_{20} & o_{21} & o_{22} \end{bmatrix}$.

It is guaranteed that:

- $|x_i| \le 10^6$, $|y_i| \le 10^6$.
- $|a_i| \le 10^3$, $|b_i| \le 10^3$, $b_i \ne 0$, $|c_i| \le 10^6$.
- All matrix elements are from 0 to $10^9 + 6$ inclusive.
- For all $1 \le i \le m$ and $1 \le j \le n$, $a_i x_j + b_i y_j \ne c_i$.
- For all $1 \le i \le m$ and $1 \le j \le m$, $\left(\frac{a_i}{b_i}, \frac{c_i}{b_i}\right) \ne \left(\frac{a_j}{b_j}, \frac{c_j}{b_j}\right)$.

Output

For each query, output a single line containing three integers s_0, s_1, s_2, indicating $\mathbf{s} = \begin{bmatrix} s_0 \\ s_1 \\ s_2 \end{bmatrix}$.

Example

standard input	*standard output*
5 1 1 2 3 4 12 12 4 6 1 1 12 5 1 2 12 1 1 5 5 6 6 2 0 3 3 1 1 4 1 1 2 3 4 5 2 3 4 1 1 400 1 3 4 2 1 2 3 4 5 -1 -1 -10 3 2 1 4 6 5 4 3 2	2 3 4 25 50 40 92 58 139

Note

Note that the solution does not depend on other properties of matrix addition/multiplication than those mentioned in the statements. Defining D and O as sets of matrices is only for testing convenience (since we can't use the graders or interaction libraries).

Problem 71. Kitten's Computer

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

Kitten recently planned to build a computer of his own. The computer has 400 registers, each of which can store a 64-bit binary integer, that is, an integer in the range $[0, 2^{64} - 1]$. The value stored in the i-th ($i \in [1, 400]$) register is denoted as a_i. This computer supports 7 assembly instructions:

- `SET i j` : Let $a_i := a_j$.
- `XOR i j k` : Let $a_i := a_j \oplus a_k$ ($\oplus$ is the bitwise XOR operation).
- `AND i j k` : Let $a_i := a_j \,\&\, a_k$ (& is the bitwise AND operation).
- `OR i j k` : Let $a_i := a_j \mid a_k$ (| is the bitwise OR operation).
- `NOT i j` : Let $a_i := \sim a_j$ ($\sim$ is the unary bitwise NOT operation).
- `LSH i x` : Shift a_i left by x bits. The vacant bit-positions are filled with 0.
- `RSH i x` : Shift a_i right by x bits. The vacant bit-positions are filled with 0.

Note that you have to ensure that $1 \le i, j, k \le 400$ and $0 \le x < 64$.

You may think that this computer is not powerful enough, but the kitten's computer is not an ordinary computer! This computer has a powerful parallel computing method that can compute all non-interfering instructions simultaneously.

Formally, let us track $t_1, t_2, \ldots, t_{400}$, denoting the times when the register values were assigned. Initially, all t_i are zeroes. Whenever you execute a command, if it requires $a_{j_1}, a_{j_2}, \ldots, a_{j_n}$ as arguments to calculate, and outputs the result to a_i, then assign t_i to $\max\{t_{j_1}, t_{j_2}, \ldots, t_{j_n}\} + 1$. The *runtime* of your program is the maximum value of all t_i generated during the sequential execution of all instructions.

Today, Kitten wants to use his computer to design a calculator. This calculator is used to quickly calculate the multiplication of 64-bit unsigned integers. At the beginning, registers a_1 and a_2 are set to two 64-bit unsigned integers x and y, respectively, while the other registers are set to 0. You need to help Kitten design a series of instructions for his program so that the final value of a_1 is the result of multiplying x and y, modulo 2^{64}.

Kitten requires that the total number of your instructions does not exceed 100 000, and the *runtime* of your program does not exceed 70.

Input

There is no input for this problem.

Output

Output any number of lines (from 0 to 100 000), each containing exactly one instruction formatted as shown above.

Example

standard input	*standard output*
<no input>	NOT 2 1 RSH 2 63 NOT 3 1 RSH 3 62 NOT 4 1 RSH 4 61 LSH 2 1 LSH 3 9 LSH 4 3 OR 5 2 3 OR 1 5 4

Note

The example output does not solve the problem, it is given only to demonstrate the format.

When checking your output, the checker will perform the following checks.

1. If your output exceeds 100 000 lines, return WA and exit immediately.
2. If your output contains an illegal instruction, return WA and exit immediately.
3. Perform the following process 5000 times:
   (a) Given are two 64-bit unsigned integers x and y.
   (b) Clear all registers to zero and make $a_1 = x$ and $a_2 = y$.
   (c) Execute your program.
   (d) If the *runtime* exceeds 70, return WA and exit immediately.
   (e) Check if the value of a_1 is $(x \cdot y) \bmod 2^{64}$. If not, return WA and exit immediately.
4. Return OK and exit immediately.

Note that the checker will only check the register a_1. The final values of all other registers can be arbitrary.

The 5000 pairs of x and y for the checker are fixed in advance.

Problem 73. Long: WCWBTT

Input file:	*standard input*
Output file:	*standard output*
Time limit:	15 seconds
Memory limit:	1024 mebibytes

This is an interactive problem.

You need to maintain a rooted tree with vertex 1 as the root. The tree has n vertices, and the parent of vertex i ($2 \le i \le n$) is p_i ($1 \le p_i < i$).

You have to process q queries, each in one of the following forms:

- "`?` a b": Output the set S, where S is the set consisting of all vertices on the unique simple path from a to b.
- "`=` a b": Change the parent of a to b (that is, $p_a \leftarrow b$). It is guaranteed that the vertices still form a tree after the modification, but it is **not guaranteed that** $b < a$.

But you soon discover a problem: the size of the set S may be too large, you can't output all elements in each query.

To deal with this issue, you have designed a special computer. This special computer can maintain sets of integers and operate on them quickly. Initially, the computer has only n+1 sets $S_0, S_1, \ldots, S_n$ where the set $S_0 = \varnothing$, and $S_i = \{i\}$ for all $1 \le i \le n$.

This computer is efficient and at the same time very simple: it supports only two different operations!

- "`+` a b": Construct a new set $S_c = S_a \cup S_b$ ($S_a \cap S_b = \varnothing$), with c being the maximum of the ID of all sets plus one. **You have to make sure that** $S_a \cap S_b = \varnothing$**.** The cost of this operation is $|S_a| + |S_b|$.
- "`!` k x_1 x_2 $\ldots$ x_k": Print the set $S_{x_1} \cup S_{x_2} \cup \cdots \cup S_{x_k}$ as the answer to the query. **You need to ensure that** $S_{x_i} \cap S_{x_j} = \varnothing$ **for all** $1 \le i < j \le k$**.** The cost of this operation is k.

Now, you need to use this computer to maintain the rooted tree. In order to avoid calculations consuming too much time and causing damage to the computer, there are the following restrictions when using the computer.

- The cost of **each operation** cannot exceed 7000.
- The **sum** of the cost of all operations cannot exceed $7.5 \cdot 10^7$.
- The total number of operations cannot exceed $5 \cdot 10^6$.

Input

The first line of the input contains two integers n and q ($1 \le n \le 2 \cdot 10^5$, $1 \le q \le 2.5 \cdot 10^4$).

The next line of the input contains $n - 1$ integers $p_2, p_3, \ldots, p_n$ ($1 \le p_i < i$), indicating the initial parent of each vertex.

Interaction Protocol

The interaction library will then perform q queries. Each query will contain one line in either the form "`?` a b" or the form "`=` a b". These forms are described above. After each query, you can perform as many operations as you want.

In particular, you need to perform operation "`!` k $\ldots$" exactly once after the query "`=` a b". After each operation "`!` k $\ldots$", you need to flush your output.

Please note that, due to the huge output, we recommend you to flush the output **only after** each operation "`! k ...`".

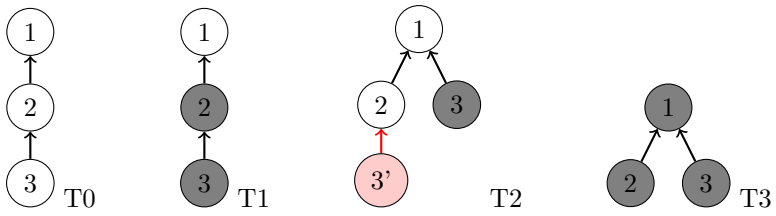
Example

standard input	*standard output*
3 3 1 2 ? 2 3 = 3 1 ? 2 3	 + 1 2 + 3 4 ! 2 2 3 <flush the output> + 1 2 + 1 3 ! 2 2 7 <flush the output>

Note

The sample input and output are intended only to illustrate the interaction protocol. The string "`<flush the output>`" and the blank lines are only added for the reader's convenience. You should not output this information.

Here's the figure of the sample test case:



The figure corresponds to the sample test case

- $S_0 = \varnothing$
- $S_1 = \{1\}$
- $S_2 = \{2\}$
- $S_3 = \{3\}$
- $S_4 = \{1, 2\}$
- $S_5 = \{1, 2, 3\}$
- $S_6 = \{1, 2\}$
- $S_7 = \{1, 3\}$

Problem 79. Matrix Counting

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	1024 mebibytes

We call an $n \times n$ matrix containing only `0`s and `1`s *bad* if and only if it contains exactly one `1` in each row and column.

Bad	Bad	Bad	Not Bad	Not Bad	Not Bad
$\begin{bmatrix}0&1\\1&0\end{bmatrix}$	$\begin{bmatrix}1&0\\0&1\end{bmatrix}$	$\begin{bmatrix}1&0&0\\0&0&1\\0&1&0\end{bmatrix}$	$\begin{bmatrix}1&1&0\\1&0&1\\0&1&1\end{bmatrix}$	$\begin{bmatrix}0&0&0\\0&1&0\\0&0&0\end{bmatrix}$	$\begin{bmatrix}0&0\\0&0\end{bmatrix}$

Define B to be a *subrectangle* of an $n \times n$ matrix A if and only if there exist $1 \le l_1 \le r_1 \le n$ and $1 \le l_2 \le r_2 \le n$ such that

- B is a $(r_1 - l_1 + 1) \times (r_2 - l_2 + 1)$ matrix.
- $B_{i,j} = A_{l_1+i-1,r_1+j-1}$ $(1 \le i \le r_1 - l_1 + 1, 1 \le j \le r_2 - l_2 + 1)$

A	B	Explanation
$\begin{bmatrix}1&0&0\\0&0&1\\0&1&1\end{bmatrix}$	$\begin{bmatrix}0&0\\0&1\end{bmatrix}$	$\begin{bmatrix}1&\mathbf{0}&\mathbf{0}\\0&\mathbf{0}&\mathbf{1}\\0&1&1\end{bmatrix}$
$\begin{bmatrix}1&0&0\\0&0&1\\0&1&1\end{bmatrix}$	$\begin{bmatrix}1&0\\0&0\end{bmatrix}$	$\begin{bmatrix}\mathbf{1}&\mathbf{0}&0\\\mathbf{0}&\mathbf{0}&1\\0&1&1\end{bmatrix}$
$\begin{bmatrix}1&0&0\\0&0&1\\0&1&1\end{bmatrix}$	$\begin{bmatrix}1&0\\0&1\end{bmatrix}$	Not a subrectangle

Given two integers n and m, you want to calculate how many $n \times n$ matrices M containing only `0`s and `1`s are there such that:

1. M is *bad*,
2. all its subrectangles of size $k \times k$ $(k = m + 1, m + 2, \ldots, n - 1)$ are not *bad*.

Since the answer can be large, output it modulo 998 244 353.

Input

The first line contains two integers n and m $(1 \le m < n \le 10^5)$.

Output

Output a single line containing a single integer, indicating the answer modulo 998 244 353.

Examples

standard input	*standard output*
3 2	6
4 2	4
300 20	368258992
100000 1	91844344

Note

In the first example, there are 6 *bad* matrices. The second condition does not matter since $m + 1 = 3 > n - 1 = 2$. So the answer is 6.

In the second example, there are 4 matrices satisfying the conditions:

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Problem 83. Puzzle: Patrick's Parabox

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Patrick's Parabox is a *Sokoban*-like game. A Sokoban puzzle is a grid, and each cell is a wall or a floor. There are several boxes and a player in some distinct floor cells, and they can not move to the wall cells or coincide. You can control the player to move in one of four directions, left, right, up, and down. When the player touches a box, it can push the box. The target is to move all boxes to some target cells.

Please read the following rules carefully. They may be different from the usual rules.

In this problem, there is **only one** box, and the box is the grid itself. That means if the player moves out of the grid, they may be "teleported" to a cell adjacent to the box; if the player moves to the box, they may be "teleported" to a cell on the boundary of the grid. Besides, there is also a target cell for the player. The player needs to move to the target cell at the end too.

Given a puzzle, you need to find the **minimum number of times to push** the box, such that the box and the player can arrive to their respective target cells.

The following are the detailed and formal rules.

Consider an $n \times m$ grid. Denote (i, j) as the cell in i-th row and j-th column. The rows are numbered $1, 2, \ldots, n$ from top to bottom, and the columns are numbered $1, 2, \ldots, m$ from left to right.

Denote W, S, A, and D as the control commands, which mean to move up, down, left, and right respectively.

Define that $v_{\mathrm{W}} = (-1, 0)$, $v_{\mathrm{S}} = (1, 0)$, $v_{\mathrm{A}} = (0, -1)$, $v_{\mathrm{D}} = (0, 1)$.

Define that $w_{\mathrm{W}} = (n, \left\lceil \frac{m}{2} \right\rceil)$, $w_{\mathrm{S}} = (1, \left\lceil \frac{m}{2} \right\rceil)$, $w_{\mathrm{A}} = (\left\lceil \frac{n}{2} \right\rceil, m)$, $w_{\mathrm{D}} = (\left\lceil \frac{n}{2} \right\rceil, 1)$.

In each operation, you can choose one of the control commands c, one of W, S, A, and D. Denote p as the cell which contains the player and b as the cell which contains the box before the operation:

- If $p + v_c = b$ and $b + v_c$ is a floor cell, the player moves to $p + v_c$ and the box moves to $b + v_c$. **Only this case** counts towards the answer, and so has to happen the minimum possible number of times.
- If $p + v_c$ is a wall cell, nothing happens.
- If $p + v_c$ is a floor cell and $p + v_c \neq b$, the player moves to $p + v_c$.
- If $p + v_c = b$ and $b + v_c$ is outside the grid, nothing happens.
- If $p + v_c = b$, $b + v_c$ is a wall cell and w_c is a wall cell, nothing happens.
- If $p + v_c = b$, $b + v_c$ is a wall cell and w_c is a floor cell, the player moves to w_c.
- if $p + v_c$ is outside the grid and $b + v_c$ is a wall cell, nothing happens.
- if $p + v_c$ is outside the grid and $b + v_c$ is outside the grid, nothing happens.
- if $p + v_c$ is outside the grid and $b + v_c$ is a floor cell, the player moves to $b + v_c$.

Note that the above are listed for covering all possibilities, but the operations are valid in only four of them (in other cases, nothing happens).

Input

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

For each test case:

The first line contains two integers n and m ($2 \leq n, m \leq 10^5$), the size of the parabox.

Each of the following n lines contains a string of length m. Each character is one of "`#`", "`.`", "`p`", "`b`", "`=`" and "`-`". Here, "`#`" denotes a wall cell, "`p`" denotes the floor cell which contains the player, "`b`" denotes the

floor cell which contains the box, "=" denotes the target floor cell of the player, "-" denotes the target floor cell of the box, and "." denotes other floor cells.

It is guaranteed that each of "p", "b", "=", "-" occurs exactly once.

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

Output

For each test case:

If it is impossible to move the box and the player to their respective target cells, output -1. Otherwise, output an integer: the **minimum number of times to push** the box.

Examples

standard input	*standard output*
3 9 9 ######### #####..-# #..=##.## #.p.##.## ....##.## #...b..## #...##.## #....#### ####.#### 9 9 ######### #.......# #.#####.# #.#=....# ..#....-# ###.p.#.# #.....#b# #.....#.# ####.#### 9 9 ####.#### #....#### #.####.## #.......# #.......# ###.##### #=.b#..## #-..p..## #########	7 4 19
1 2 2 pb -=	-1

Note

The three puzzles in the first example are real levels in the game.

Problem 89. Poker Game: Construction

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Alice and Bob invent a new game based on Texas hold'em. **Please read the following rules carefully as they are different from the usual rules. The background of this problem is exactly the same as problem D.**

There are 13 ranks, which are `A`, `K`, `Q`, `J`, `T`, `9`, `8`, `7`, `6`, `5`, `4`, `3`, and `2` from high to low. There are 4 suits, which are `S`, `H`, `C`, and `D`. Every combination of a rank and a suit occurs exactly once, so there are $52 (= 13 \times 4)$ cards.

A hand is a set of five cards. Each hand has a rank. There are 10 types of hands. Each type also has a rank. If two hands are of different types, the hand of the type with a higher rank always ranks higher. A hand can be represented as a sequence $(r_1, r_2, r_3, r_4, r_5)$, where r_i is the rank of the i-th card and the order of the five cards depends on the type of the hand. If two hands are of the same type, the hand represented as the lexicographically larger sequence ranks higher: formally, if we find the smallest index i such that r_i of two hands are different, the hand with higher r_i ranks higher. If the types and the sequences r of two hands are equal, two hands have the same rank.

The 10 types are given below, from lowest to highest rank. If a hand matches the patterns of multiple types, it belongs to the one with the highest rank.

- **Highcard**: Any five cards. The sequence r satisfies $r_1 > r_2 > r_3 > r_4 > r_5$.
- **Pair**: Two cards with the same rank. The sequence r satisfies $r_1 = r_2$, $r_3 > r_4 > r_5$.
- **Two pairs**: Two cards with the same rank and another two cards with the same rank. The sequence r satisfies $r_1 = r_2 > r_3 = r_4$.
- **Three of a kind**: Three cards with the same rank. The sequence r satisfies $r_1 = r_2 = r_3$, $r_4 > r_5$.
- **Straight**: Five cards with five consecutive ranks. The sequence r satisfies $r_1 > r_2 > r_3 > r_4 > r_5$. Additionally, `A 2 3 4 5` is a straight, and `A` is regarded as a rank lower than `2` in this case. Hence `A 2 3 4 5` is the straight with the lowest ranks.
- **Flush**: Five cards with the same suit. The sequence r satisfies $r_1 > r_2 > r_3 > r_4 > r_5$.
- **Full house**: Three cards with the same rank and another two cards with the same rank. The sequence r satisfies $r_1 = r_2 = r_3$, $r_4 = r_5$.
- **Four of a kind**: Four cards with the same rank. The sequence r satisfies $r_1 = r_2 = r_3 = r_4$.
- **Straight flush**: A straight with the same suit. The sequence r satisfies $r_1 > r_2 > r_3 > r_4 > r_5$. Additionally, `A 2 3 4 5` with the same suit is a straight flush, and `A` is regarded as a rank lower than `2` in this case. Hence `A 2 3 4 5` with the same suit is the straight flush with the lowest ranks.
- **Royal flush**: Straight flush with the ranks `T`, `J`, `Q`, `K`, and `A`. Four different royal flushes are of the same rank.

Two cards are dealt to each of Alice and Bob. Instead of the regular rules, 6 community cards are dealt. Two players take community cards one by one, in turn, until each player has five cards, completing a hand. **Alice takes first.** The player who has a hand with higher rank wins. If two hands have same rank, there is a draw. **Note that all ten cards are shown to both, and they always choose the optimal strategy.**

The above are the same in problem D. The task is the following.

Given the cards of Alice and the cards of Bob, find possible 6 community cards such that Alice wins, that Bob wins, and that there is a draw.

Input

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

The first line contains two strings a_1 and a_2: Alice's initial cards.

The second line contains two strings b_1 and b_2: Bob's initial cards.

Each string is of length two. The first character is one of "`A`", "`K`", "`Q`", "`J`", "`T`", "`9`", "`8`", "`7`", "`6`", "`5`", "`4`", "`3`", "`2`", which represents the rank of a card. The second character is one of "`S`", "`H`", "`C`", "`D`", which represents the suit of a card.

It is guaranteed that all 4 given cards are pairwise distinct.

Output

For each test case:

Print a single line for each of the three questions: how to make Alice win, how to make Bob win, and how to make the game end in a draw. For each question, if it is possible, output "`YES`" and 6 strings representing the 6 cards; otherwise, output "`NO`".

The format of the cards in the output should be the same as the format in the input.

All 10 cards in input and output must be pairwise distinct.

Examples

standard input	*standard output*
3 JC 4H TS 5D 7C 3C 7H TH 2D KH 4D JC	YES JS JH JD 4S 4C 4D YES TH TC TD 5S 5H 5C YES 4S JS 5S TH TC TD YES 3S 3H 3D 2C 4H 5S YES 2H 4H 5H 6H 8H 9H YES TS 3S 2S 2H 4C 4D YES 2S 2H 2C KS KC KD YES 4S 4H 4C JS JH JD YES 2S KS 4S JS JH JD
2 AS AH AC AD AS AH 2S 2H	YES 2H 3S 4D 5C 6H 7S NO YES 2C 2D 3S 3H 4C 4D YES AC AD 3D 4C 5H 6S YES 2C 2D 3D 4C 5H 6S NO

Problem 97. Symmetry: Closure

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

A point set S is symmetric about a line ℓ if and only if there exists $s' \in S$ satisfying that s' and s are symmetric about the line ℓ for all $s \in S$.

Let us denote the distance between two points a and b as $d(a, b)$. The distance between two non-empty point sets A and B is $\inf\{d(a, b) : a \in A, b \in B\}$. The infimum of a non-empty real number set S is the maximum value of x which satisfies $x \leq s$ for all $s \in S$.

Lines $\ell_1, \ell_2, \ldots, \ell_n$ are given, where two or more lines may coincide. For a point s, define $C(s)$ as the intersection of all sets S satisfying $s \in S$ such that S is symmetric about ℓ_i for all $i = 1, 2, \ldots, n$.

There are q queries. For each query, given two points A and B, find the distance between $C(A)$ and $C(B)$.

Input

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

The first line contains an integer n and q ($1 \leq n, q \leq 10^5$): the number of lines and the number of points.

The i-th of the following n lines contains four integers x_{P_i}, y_{P_i}, x_{Q_i}, and y_{Q_i}: the coordinates of P_i and Q_i such that ℓ_i passes through P_i and Q_i. It is guaranteed that $x_{P_i} \neq x_{Q_i}$ or $y_{P_i} \neq y_{Q_i}$. Any two lines may coincide.

The i-th of the following q lines contains four integers x_{A_i}, y_{A_i}, x_{B_i}, and y_{B_i}: the coordinates of A_i and B_i.

It is guaranteed that the absolute value of all coordinates in the input does not exceed 10^9.

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

Output

For each test case:

For each query, output the distance between $C(A)$ and $C(B)$.

The distance you output will be considered correct if the relative error or absolute error to the jury does not exceed 10^{-9}.

Examples

standard input	*standard output*
4 1 1 0 0 1 0 -1 -2 2 1 2 1 0 0 1 0 0 0 0 1 -1 -2 2 1 3 1 0 0 1 0 0 0 0 1 0 0 1 1 -1 -2 2 1 3 1 0 0 1 0 0 0 0 1 0 0 1 2 -1 -2 2 1	3.162277660168 1.414213562373 0.000000000000 0.000000000000
5 1 1 -8 1 -8 10 -7 -5 -4 -6 2 2 -1 -10 -1 -8 10 9 9 10 2 10 -10 5 -4 4 -3 -3 3 1 -5 -10 -5 6 6 10 8 8 7 -2 4 -5 0 -9 -6 -3 3 3 9 8 10 7 1 5 -9 5 4 -2 -3 -9 6 6 -6 -8 2 -7 10 -3 3 -8 8 -9 1 3 10 -9 10 -7 -2 -7 -2 6 -2 9 -9 2 -6 -7 -7 -9	3.162277660168 7.810249675907 7.071067811865 7.211102550928 0.000000000000 0.000000000000 0.000000000000 13.000000000000 9.899494936612 2.236067977500

Problem 101. Noodle

Input file:	*standard input*
Output file:	*standard output*
Time limit:	7 seconds
Memory limit:	1024 mebibytes

Putata is a boy who loves eating noodles. Now he's waiting for the great chef Budada to cook the most delicious noodle ever for him.

The noodle which Budada is cooking for him can be described as an array a of length n, where n is even. The amount of sauce initially at position i is a_i.

In one operation, Budada will do the following process.

1. Budada will fold the noodle, the length of the noodle will become $\frac{n}{2}$, the amount of sauce at position i will become the sum of the amounts of sauce at positions i and $n-i+1$. Formally, the amount of sauce at position i of the new noodle b_i satisfies $b_i = a_i + a_{n-i+1}$.
2. Then Budada will stretch the noodle to the original length, and the amount of sauce will be evenly divided. Formally, the amount of sauce at position i of the new noodle a'_i satisfies $a'_i = \frac{1}{2} \cdot b_{\lceil \frac{i}{2} \rceil}$.

Putata has a favorite position on the noodle, which is a certain position x. Now you are asked to answer q queries. In the i-th query, you should output the amount of sauce at position x after k operations. The x is the same for all queries, but k is given separately for each query.

It can be shown that the answer can be expressed as an irreducible fraction $\frac{x}{y}$, where x and y are integers and $y \not\equiv 0 \pmod{998\,244\,353}$. Output the integer equal to $x \cdot y^{-1} \pmod{998\,244\,353}$. In other words, output such an integer a that $0 \le a < 998\,244\,353$ and $a \cdot y \equiv x \pmod{998\,244\,353}$.

Since the input is quite large, you will have to use a generator to generate the queries, and you only have to output $\oplus_{i=1}^{q}(ans_i \cdot i)$. Please notice that this number is **not** taken modulo $998\,244\,353$. Here, $\oplus$ means bitwise exclusive-or operation.

Input

The first line contains three integers *test*, T, and *seed*, which are an **unrelated variable**, the number of test cases, and the seed for generating test data. Please note that *test* **will not** be used to solve the problem, you can just ignore it. The generator code is given further below.

For each test case, the input will contain two lines.

The first line contains four integers n, q, x, and $k_{\max}$ ($1 \le n \le 2 \cdot 10^6$, $1 \le q \le 5 \cdot 10^7$, $1 \le x \le n$, $1 \le k_{\max} \le 10^{18}$).

The second line contains n integers, the i-th integer is a_i ($0 \le a_i < 998\,244\,353$).

It is guaranteed that $\sum n \le 2 \cdot 10^6$, $\sum q \le 5 \cdot 10^7$, and n is even.

Output

Output T lines. The i-th line must contain the answer to the i-th test case.

Example

standard input	*standard output*
0 2 13 4 2 1 3 1 4 2 3 6 2 3 3 6 2 5 3 1 4	5 499122191

Note

In the first test case of the sample, $\{a_i\}$ are $\{1, 4, 2, 3\}$ initially.

- After one operation, it becomes $\{2, 2, 3, 3\}$.
- After two operations, it becomes $\{\frac{5}{2}, \frac{5}{2}, \frac{5}{2}, \frac{5}{2}\}$.
- The generated queries are:
- The position is $x = 1$;
- The first query: $k = 0$, $a_x = 1$;
- The second query: $k = 1$, $a_x = 2$;
- The answer is $(1 \cdot 1) \oplus (2 \cdot 2) = 5$.

In the second test case, $\{a_i\}$ is $\{6, 2, 5, 3, 1, 4\}$ initially.

- After one operation, it becomes $\{5, 5, \frac{3}{2}, \frac{3}{2}, 4, 4\}$.
- After two operations, it becomes $\{\frac{9}{2}, \frac{9}{2}, \frac{9}{2}, \frac{9}{2}, \frac{3}{2}, \frac{3}{2}\}$.
- The generated queries are:
- The position is $x = 3$;
- The first query: $k = 2$, $a_x = \frac{9}{2}$, and $\frac{9}{2} \equiv 499\,122\,181 \pmod{998\,244\,353}$;
- The second query: $k = 0$, $a_x = 5$.
- The answer is $(499\,122\,181 \cdot 1) \oplus (5 \cdot 2) = 499\,122\,181 \oplus 10 = 499\,122\,191$.

The generator will be given below:

```
#include <bits/stdc++.h>
using namespace std;

unsigned long long rd (unsigned long long &x) {
  x ^= (x << 13);
  x ^= (x >> 7);
  x ^= (x << 17);
  return x;
}

int main () {
  int test, T;
  unsigned long long seed;
  scanf("%d%d%llu", &test, &T, &seed);
  for (int Case = 1; Case <= T; Case ++) {
    int n, q, x;
    long long k_max;
    scanf("%d%d%d%lld", &n, &q, &x, &k_max);
    vector<int> a(n + 1);
    for (int i = 1; i <= n; i ++) {
      scanf("%d", &a[i]);
    }
    for (int i = 1; i <= q; i ++) {
      long long k = rd(seed) % k_max;
      /*
      Code your solution here.
      */
    }
  }
}
```

Problem 103. Geometry

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

Grammy has a special two-dimensional coordinate system: the angle between the positive half-axis of the X-axis and the positive half-axis of the Y-axis is 60 degrees.

Consider the following graph. The vertices are all integer coordinates (x, y) such that at least one of x, y is odd and $-2a+1 \le x \le 2a-1$, $-2b+1 \le y \le 2b-1$, $-2c+1 \le x+y \le 2c-1$. The edges from (x, y) go to $(x, y+1)$, $(x, y-1)$, $(x+1, y)$, $(x-1, y)$, $(x+1, y-1)$, and $(x-1, y+1)$.

Find the size of the maximum independent set of vertices in this graph. Additionally, find the number of such sets modulo 998 244 353.

Input

The first line contains an integer T ($1 \le T \le 10$), denoting the number of test cases.

Each of the following T lines contains three integers a, b, c ($1 \le a, b, c \le 10^6$).

Output

Output T lines. Each line must contain two integers: the size of the maximum independent set and the number of such sets. **Please note that the size should not be taken modulo** 998 244 353.

Example

standard input	*standard output*
6 2 1 2 1 1 137 3 94 95 3 1998 1996 998244 353999 999999 50 120 150	7 4 4 1 1124 31585548 23951 33873190 1289433675488 748596399 23600 480090154

Note

The following picture shows the situation for the first and second test case of the sample.

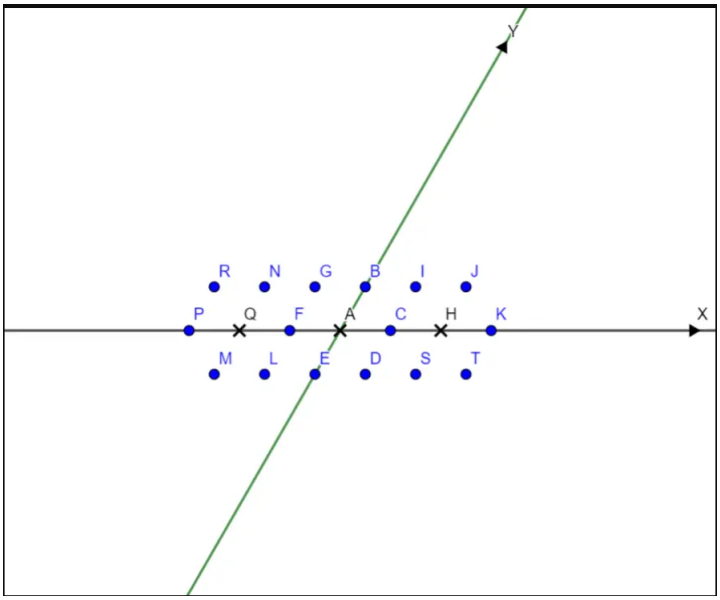
Point J has coordinates $(2, 1)$, point F has coordinates $(-1, 0)$, and point H has coordinates $(2, 0)$. Among these three points, only H has even X-coordinate and even Y-coordinate. The neighbours of point A are $BCDEFG$.

In the first test case, the points that satisfy the conditions are $NGBIJPFCKMLEDST$.

The size of the maximum independent set is 7, and there are 4 ways: $PNLBDJT$, $RMFBDJT$, $RMGECJT$, $RMGEISK$.

In the second test case, the points that satisfy the conditions are $GBIFCLED$.

The size of the maximum independent set is 4, and there is one way: $LGID$.



Picture for test case 1 and 2.

Problem 107. DFS

Input file:	*standard input*
Output file:	*standard output*
Time limit:	8 seconds
Memory limit:	1024 mebibytes

You are given a rooted tree of n vertices, and r is the root of the tree. Each vertex x has value a_x.

Let us define the DFS procedure starting from x to find y:

1. Push x on the stack.
2. Check w, the top element of the stack. If $w = y$, the procedure ends. Otherwise, if there is at least one son of w which is not visited, choose one such son with equal probability and push it on the stack.
3. Repeat step 2 until there is no unvisited son.
4. Pop the top element from the stack.
5. Repeat step 2 until the stack is empty.

The procedure is legal if and only if y is in the subtree of x.

Define $f(x, y)$ as the expectation of the minimum value of all vertices which were pushed on the stack during the DFS procedure starting from x to find y.

Now we want to calculate $\sum f(x, y)$ for all legal pairs (x, y). It can be shown that the answer can be expressed as an irreducible fraction $\frac{x}{y}$, where x and y are integers and $y \not\equiv 0 \pmod{998\,244\,353}$. Output the integer equal to $x \cdot y^{-1} \pmod{998\,244\,353}$. In other words, output an integer a such that $0 \le a < 998\,244\,353$ and $a \cdot y \equiv x \pmod{998\,244\,353}$.

Input

The first line contains an integer T ($1 \le T \le 100$), denoting the number of test cases.

For each test case, the first line contains two integers n and r ($1 \le n \le 4 \cdot 10^5$, $1 \le r \le n$), denoting the number of vertices in the tree and the root.

The following line contains n integers, the i-th integer of them is $a_i (1 \le a_i \le 10^9)$ denoting the value of vertex i.

Each of the next $n - 1$ lines contains two integers u and v ($1 \le u, v \le n$), denoting an edge of the tree.

It is guaranteed that $\sum n \le 8 \cdot 10^5$. It is also guaranteed that the given graph is indeed a tree.

Output

Output T lines. Each line must contain one integer: the answer to the respective test case.

Example

standard input	*standard output*
4 1 1 1 3 3 3 3 4 3 1 3 2 6 1 5 2 4 1 3 6 1 2 1 6 2 3 2 4 4 5 5 1 5 4 3 2 1 1 2 1 3 3 4 3 5	1 16 34 499122202

Problem 109. Vacation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	1024 megabytes

Prof. Pang has an annual leave of c days and he wants to go on vacation.

Now there are n days in a year. Prof. Pang can gain a_i happiness if he rests on the i-th day. The values of happiness, a_i, may be negative.

Prof. Pang wants you to do m operations:

- 1 x y, change the happiness of the x-th day to y.
- 2 l r, Prof. Pang wants to find a period of vacation in $[l, r]$. He wants to rest for several (possibly 0) days in a row and gain as much happiness as possible. However, he only has c days off, thus he can rest for no more than c consecutive days in $[l, r]$.

That means he wants to find

$$\max\left(\max_{\substack{l\le l'\le r'\le r\\ r'-l'+1\le c}}\left(\sum_{i=l'}^{r'} a_i\right), 0\right).$$

Input

The first line contains three integers $n, m, c (1 \le n \le 2 \times 10^5, 1 \le m \le 5 \times 10^5, 1 \le c \le n)$ indicating the number of days in a year, the number of operations, and Prof. Pang's annual leave days.

The next line contains n integers $a_1, a_2, \ldots, a_n (-10^9 \le a_i \le 10^9)$ indicating the values of happiness of every day.

The next m lines are the m operations in the format described above.

It is guaranteed that $1 \le x \le n, -10^9 \le y \le 10^9, 1 \le l \le r \le n$.

Output

For each operation of the second type, print the answer.

Example

standard input	standard output
5 6 3 0 -5 -3 8 -3 2 3 5 1 2 5 2 1 5 1 4 -3 2 3 5 2 1 5	8 10 0 5

Problem 113. Vision Test

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Prof. Pang has an extraordinary vision. He can see the pixels on a 4K monitor. To test Prof. Pang's vision, Prof. Shou will show Prof. Pang several pixels and let Prof. Pang guess a straight line that contains these pixels. Given k pixels with coordinates (i, y_i) $(0 \le i < k)$, Prof. Pang must find nonnegative integers a, b and c (which represent the line $y = \frac{ax+b}{c}$) such that $y_i = \lfloor \frac{ai+b}{c} \rfloor$ for all $0 \le i < k$.

Prof. Shou will ask Prof. Pang multiple questions. They are given as follows: Prof. Shou has a fixed array $x_1, \ldots, x_n$. For each question, Prof. Shou chooses a range in the array, $x_l, \ldots, x_r$. Then he defines $y_i = x_{l+i}$ for $0 \le i \le r-l$ and asks Prof. Pang to answer the question for the $r-l+1$ pixels $(0, y_0), \ldots, (r-l, y_{r-l})$.

Please help Prof. Pang answer all the questions. For each question, output the answer with the **minimum** (c, a, b) **in lexical order**.

It is guaranteed that the answer exists when Prof. Pang chooses the whole array $x_1, x_2, \ldots, x_n$. So the answer always exists when Prof. Pang chooses an interval of this array.

Input

The first line contains a single integer T $(1 \le T \le 10^5)$ denoting the number of test cases.

For each test case, the first line contains an integer n $(1 \le n \le 10^5)$. The second line contains n numbers $x_1, \ldots, x_n$ $(0 \le x_i \le 10^9)$.

The next line contains an integer q $(1 \le q \le 10^5)$ denoting the number of questions.

Each of the following q lines contains two integers l, r $(1 \le l \le r \le n)$.

It is guaranteed that the sum of n over all test cases will not exceed 10^5 and that the sum of q over all test cases will not exceed 10^5.

Output

In the order of input, output one line with three integers a, b, c denoting the answer for each question.

Example

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

Problem 127. Sequence to Sequence

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 megabytes

Chiaki has a sequence $s_1, s_2, \ldots, s_n$. She would like to change it to another sequence $t_1, t_2, \ldots, t_n$ using the following operations:

- choose two indices l and r ($l \le r$), and add 1 to every nonzero element between the indices l and r (both inclusive).
- choose two indices l and r ($l \le r$), and subtract 1 from every nonzero element between the indices l and r (both inclusive).

Chiaki would like to know the minimum number of operations needed.

Input

There are multiple test cases. The first line of input contains an integer T, indicating the number of test cases. For each test case:

The first line contains an integer n ($1 \le n \le 10^5$) – the length of the sequence.

The second line contains n integers $s_1, s_2, \ldots, s_n$ ($0 \le s_i \le 10^9$).

The third line contains n integers $t_1, t_2, \ldots, t_n$ ($0 \le t_i \le 10^9$).

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

Output

For each test case, output an integer denoting the minimum number of operations. If it is impossible to change the sequence, output -1 instead.

Example

standard input	standard output
2 5 1 1 1 1 1 2 0 2 0 2 7 3 1 2 3 2 1 4 2 0 0 0 0 0 2	3 3

Note

For the first test case: $\{1,1,1,1,1\} \xrightarrow{[2,2],\ -1} \{1,0,1,1,1\} \xrightarrow{[4,4],\ -1} \{1,0,1,0,1\} \xrightarrow{[1,5],\ +1} \{2,0,2,0,2\}$.

Problem 131. Keychain

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	256 mebibytes

Consider a two-dimensional plane and n points $p_1, \ldots, p_n$ on it. Consider n circles $C_1, C_2, \ldots, C_n$: the i-th circle is centered at p_i. All the radii of the n circles are R.

Determine the minimum value of R such that one can draw another *generalized circle* Γ that intersects all the n circles. Please find one such Γ as well.

- A circle C with radius r contains all points such that the Euclidean distance between the point and the center of the circle is exactly r.
- A *generalized circle* is either a circle or a straight line.
- We say two objects A and B intersect if they share a common point.

Input

The first line contains an integer n ($1 \le n \le 3000$). On each of the next n lines, there will be two integers x_i and y_i indicating the coordinates of point p_i ($0 \le x_i, y_i \le 10^5$). It is guaranteed that no two given points coincide.

Output

On the first line, print the optimal answer R_{opt}.

Your output should satisfy $0 \le R_{opt} \le 10^5$.

It can be proved that the minimum value exists and is in this range.

Suppose that Γ_{opt} intersects all $C_1, \ldots, C_n$ when $R = R_{opt}$.

It can be shown that, under the constraints in this problem, Γ_{opt} can be chosen to be either a circle centered at a rational coordinate, or a straight line with integer coefficients.

- In the circle case, print "`C` X Y Z r", which means that the radius is r, and the center of the circle is $O = (X/Z, Y/Z)$.

  The values X, Y, Z must be integers with absolute value not greater than 10^{18}. The value r should be a non-negative real number not greater than 10^{18}.

- In the straight line case, print "`L` a b c", which means that the line L satisfies the equation $ax + by = c$.

  The values a, b, c must be integers with absolute value not greater than 10^{18}.

When checking your answer, the jury will first check whether Γ_{opt} intersects each of the C's. This will be judged by checking:

- if $|R - r| - \varepsilon \le d(O, p_i) \le R + r + \varepsilon$ in the circle case ($d(O, p_i)$ is the Euclidean distance between p_i and O),
- or $R \ge d(L, p_i) - \varepsilon$ in the line case ($d(L, p_i)$ is the distance from point p_i to line L).

Here, $\varepsilon = 10^{-6}$.

After that, your answer will be considered correct if the absolute or relative error between your R_{opt} and jury's R_{opt} doesn't exceed 10^{-6}.

Examples

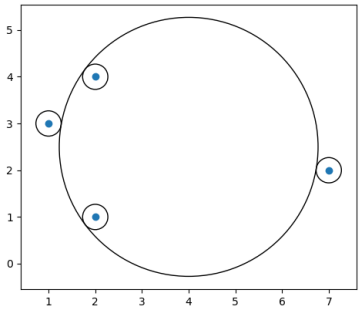
standard input
4 2 1 1 3 2 4 7 2
standard output
0.27069063257455492223 C 1152 720 288 2.77069063257455492234

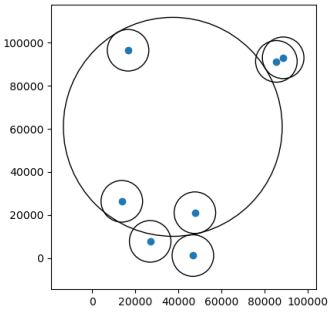
standard input
7 26919 7739 85584 91359 47712 21058 13729 26355 16636 96528 88747 93023 46770 1150
standard output
9663.87959749101919015857 C 3605577680770432 5873755742321056 96368792608 50864.33205303458045065668

standard input
10 756 624 252 208 504 416 378 312 203 287 329 391 0 0 707 703 126 104 581 599
standard output
46.05915288207108030175 L -1248 1512 90300

Note

The first two examples:





Be careful of overflow. Consider using `long double` or `__int128`.

Problem 137. Greedy Bipartite Matching

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	1024 mebibytes

Consider a bipartite weighted graph with $2n$ vertices: n in the left part and n in the right part. The vertices in each part are numbered from 1 to n. A matching is called *greedy* if it has the maximal number of edges of weight 1 among all matchings, the maximal number of edges of weight 2 among all matchings that maximize the number of edges of weight 1, etc.

Your task is to find the size (number of edges) of greedy matching in a dynamically growing graph.

Input

The first line of the input contains two non-negative integers n and q ($n \le 10^5$, $q \le 10^3$): the number of vertices in each part and the number of different weights of the edges.

Then, the input consists of q blocks. The i-th block starts with a non-negative integer m_i: the number of edges of weight i. Each of the next m_i lines contains two integers x and y ($1 \le x, y \le n$), which add an edge between vertex x of the left part and vertex y of the right part. It is guaranteed that $\sum_i m_i \le 2 \cdot 10^5$.

Note that there may be multiple edges between two vertices.

Output

You have to output q integers on a single line: answers for the problem for weights at most 1, weights at most 2, ..., weights at most q.

Example

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

Problem 139. Endless Road

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	1024 mebibytes

Suppose we have three chips on integer points on an infinite line (it is possible that two or more chips are at the same point). Every second one chip, taken equiprobably, moves to the next integer point (if the point was equal x, it becomes $x + 1$).

For each value of t from 1 to n, your task is to find the expected value of the maximal chip coordinate after t seconds.

Input

The first line of input contains three integers a, b, c ($0 \le a \le b \le c \le 10^6$): the initial coordinates of the chips.

The second line contains a single integer n ($1 \le n \le 2 \cdot 10^6$): the maximal time we are interested in.

Output

For each t from 1 to n, print a single line with a single number: the expected value of the maximal chip coordinate after t seconds, expressed as an integer modulo prime number 998 244 353. Formally, you can see that the expectation is a rational number $\frac{p}{q}$, where q is coprime with 998 244 353. You should output the number pq^{-1} modulo 998 244 353.

Examples

standard input	*standard output*
0 0 0 5	1 332748119 554580198 813384290 110916042
111 222 456 10	332748574 665496692 457 332748575 665496693 458 332748576 665496694 459 332748577

Problem 149. LCSLCSLCS

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	1024 mebibytes

Suppose we have two non-empty strings A and B ($|A|, |B| \leq 500$) of capital English letters, and two integers n and m such that $1 \leq n, m \leq 10^{15}$.

Let string A^n be a concatenation of n copies of string A. Let string B^m be a concatenation of m copies of string B. Your task is to find the longest common subsequence of A^n and B^m.

Input

On the first line, there are two integers n and m ($1 \leq n, m \leq 10^{15}$).

On the second line, there is a non-empty string A with length at most 500.

On the third line, there is a non-empty string B with length at most 500.

Both strings consist of capital English letters.

Output

Output one integer: the length of the longest common subsequence of A^n and B^m.

Examples

standard input	*standard output*
10 10 AB BA	19
100000000 100000000 A AB	100000000

Problem 151. Colonization

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	512 megabytes

Recently, Bytecja has begun the colonization of a newly discovered continent and its countless surrounding islands. Unfortunately, the indigenous people are not pleased with the arrival of the colonizers.

The colonization process on an island is as follows. There are n villages on the island, connected by $n-1$ bidirectional roads in such a way that you can get from any village to any other. In other words, the layout of villages and roads on each island forms a tree. Initially, in a village chosen by the colonizers, k colonizers appear, making this village colonized. Then, the colonizers can freely move along the roads or wait for the movements of other colonizers. A village is colonized the moment a colonizer arrives.

Taking over the whole island would be simple even with one colonizer, but there's a catch – if a colonized village is left without a waiting colonizer and it neighbors an uncolonized village, the locals might raid it, leading to a tragic outcome. So, this situation must be avoided. A colonizer has to guard such a village or be on the road between these villages, as the locals will definitely not bypass him.

The question for a given setup is the minimum number of k for which there's a choice of the initial village and a strategy for the colonizers' movements that allows for the whole island to be colonized.

Your task is to determine, given the number n, for each k, the possible road layouts on the island which require exactly k colonizers to be conquered. Two road layouts are considered different if the trees they represent are not isomorphic (meaning you need to count **unlabeled trees**). In other words, two road layouts are different if there is no bijection between villages in one layout and the villages in the other layout such that two villages in the first layout are connected by a road if and only if their corresponding villages in the second layout are connected by a road.

Since these numbers can be very large, you can provide the modulus of these numbers by a given prime.

Input

The first and only line of the standard input contains two integers n and p ($2 \le n \le 500$; $10^8+7 \le p \le 10^9+7$; p is a prime number), representing the considered number of villages on the island and the mentioned prime number, respectively.

Output

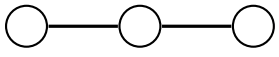
The first and only line of the standard output should contain n integers. The k-th number should represent the number of different road layouts among the n villages that require exactly k colonizers to be conquered, provided modulo p.

Examples

standard input	standard output
3 100000007	1 0 0
6 300000007	1 5 0 0 0 0
10 1000000007	1 104 1 0 0 0 0 0 0 0

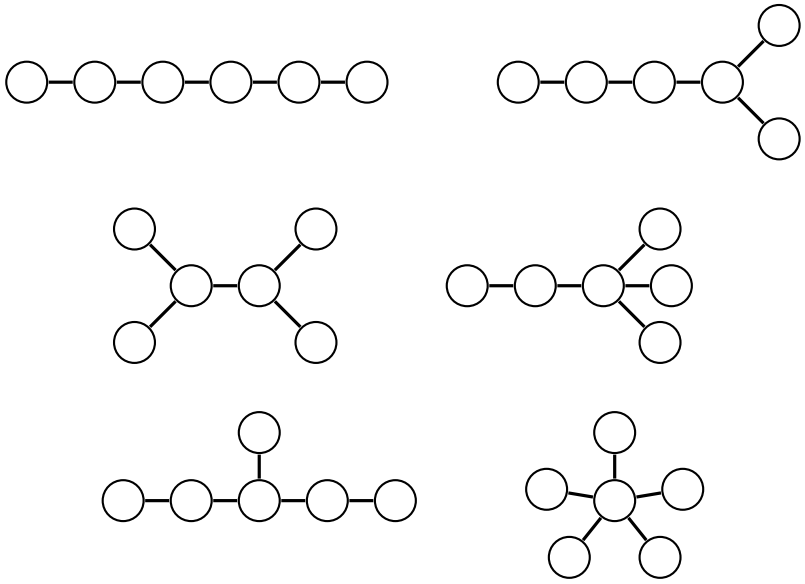
Note

In the first sample test, there is only one possible road layout and it looks as follows:



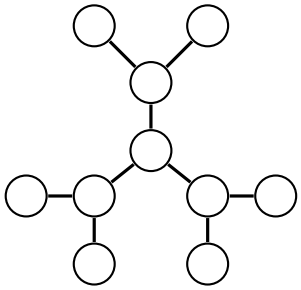
To colonize such an island, only one colonizer is needed, provided he doesn't start in the central village.

In the second sample test, there are 6 possible road layouts and they look as follows:



Only the top-left of these layouts can be colonized with the help of just one colonizer. For all the others, exactly two of them are required. For example, in the middle-right layout, they can start in the far-left village, move together two villages to the right, and then one of them will wait for the other, who will then consecutively colonize the remaining three villages.

In the third sample test, there is only one layout that requires three colonizers and it looks as follows:



Problem 157. Redundant Towers

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	1024 megabytes

There are n wireless communication towers in Byteland, labeled by $1, 2, \ldots, n$. The i-th tower is located at (x_i, y_i). No two towers share the same x-coordinate, and no two towers share the same y-coordinate. The transmission radius of each tower is R. In other words, the i-th tower can send a message to the j-th tower in one jump if and only if $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \leq R$. Initially, all the towers are in use.

The a-th ($1 \leq a \leq n$) tower is considered to be redundant if and only if it satisfies all the following conditions:

- The a-th tower is in use.
- For every pair of towers b and c ($1 \leq b < c \leq n$, $b \neq a$, $c \neq a$), if they are both in use, and b can send message directly or indirectly to c, then b can also send message directly or indirectly to c without passing through the a-th tower. Note that messages can only be transmitted by towers in use.

You are required to perform q operations. In each operation, you will be given an integer k ($1 \leq k \leq n$), denoting the label of the tower whose status is changed. If the k-th tower is in use, it will now become not in use, and if it's not in use, it will now become in use. After each operation, you are required to count the number of redundant towers.

Input

The first line contains two integers n and R ($1 \leq n \leq 10^5$, $2 \leq R \leq 5$), denoting the number of towers and the transmission radius.

In the next n lines, the i-th line contains two integers x_i and y_i ($1 \leq x_i, y_i \leq n$), denoting the location of the i-th tower. It is guaranteed that no two towers share the same x-coordinate, and no two towers share the same y-coordinate.

The next line contains a single integer q ($1 \leq q \leq 10^5$), denoting the number of operations.

Each of the next q lines contains a single integer k ($1 \leq k \leq n$), denoting an operation. Let *last* be the previous number of redundant towers that you answered. Note that *last* should be reset to 0 at the beginning of the input. For each operation, k is encrypted: its actual value is $k \oplus last$. In the expressions above, the symbol "$\oplus$" denotes the bitwise exclusive-or operation. Also, note that the constraints described in the statement above apply to the corresponding parameters only after decryption, the encrypted values are not subject to those constraints.

Output

For each operation, print a single line containing a single integer, denoting the number of current redundant towers.

Example

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

Problem 163. Rook Detection

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	512 megabytes

This is an interactive problem.

Consider an $n \times n$ chessboard, where its rows are numbered 1 to n from top to bottom, and its columns are numbered 1 to n from left to right. The square on the x-th row and the y-th column is denoted as (x, y) and is either empty or occupied by a neutral rook. According to the rules of chess, a rook can move any number of squares along a row or column in one step but cannot leap over other pieces. In this scenario, the *control area* of a rook consists of the square it occupies and the squares it can reach in one step.

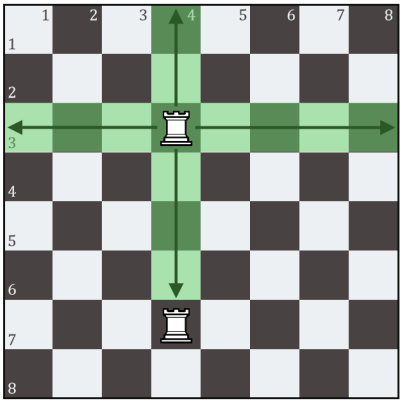


Figure 1: The control area of the rook at (3,4)

A square is considered "*controlled*" if and only if it belongs to one or more control areas. The known information is that all squares are controlled initially, so clearly, there are at least n rooks on the chessboard. Whether each square is empty or occupied by a rook, however, is unknown to you. Your task is to locate at least n of the hidden rooks by making no more than $\lceil \log_2 n \rceil + 2$ queries.

In each query, you need to split the chessboard into two tiers by determining which squares should comprise the upper tier and which the lower tier. The rooks are only allowed to move within the tier to which they belong, and they can leap over gaps but cannot leap over other pieces in the same tier. The definitions of control areas and controlled squares do not change. As a response, the interactor will inform you whether each square is controlled after the splitting. Then, the chessboard will return to the initial state.

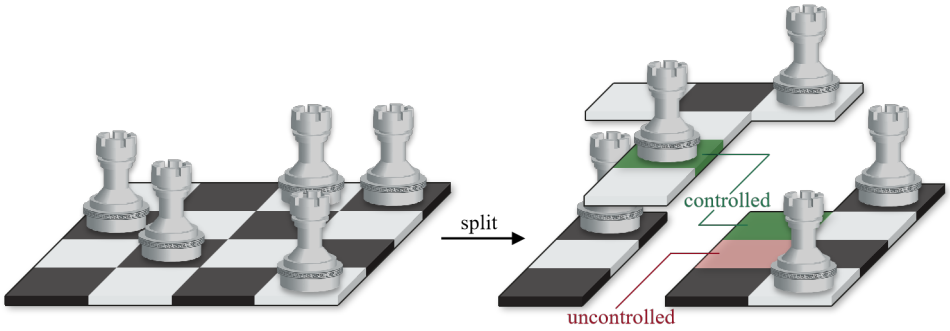


Figure 2: Split the T-shaped upper tier out of the chessboard

You may assume that the locations of all the rooks are fixed in advance. Note again that it is not necessary to find all the rooks when the actual number of rooks is greater than n.

Interaction Protocol

The first line contains an integer T ($1 \le T \le 10^4$), denoting the number of test cases.

The first line of each test case contains an integer n ($3 \le n \le 500$), denoting the size of the chessboard. It is guaranteed that the number of test cases that $n \ge 10$ does not exceed 5.

To initiate an interaction behavior, you should firstly print a character *opt* (*opt* $\in$ {'?', '!'}) in one line and then print an $n \times n$ binary matrix (in n lines, each containing a binary string). After printing the binary matrix, do not forget to flush the output. To do this, use `fflush(stdout)` or `cout.flush()` in C++, `System.out.flush()` in Java, or `stdout.flush()` in Python.

opt = '?' represents making a query. If the character in the x-th row and y-th column of your binary matrix is '1', the square (x, y) will comprise the upper tier, or otherwise, it will comprise the lower tier. The interactor will return an integer *code* (*code* $\in \{0, 1\}$) in one line. If *code* = 1, you will get `Wrong Answer` because you made more than $\lceil \log_2 n \rceil + 2$ queries. If *code* = 0, you will then receive another $n \times n$ binary matrix from the interactor. If the character in the x-th row and y-th column of the received binary matrix is '1', the square (x, y) is controlled after the splitting, or otherwise, it is uncontrolled.

opt = '!' represents answering the locations. If the character in the x-th row and y-th column of your binary matrix is '1', the square (x, y) is occupied by a rook in your judgment, or otherwise, whether it is empty or occupied by a rook will not affect the verification of your answer. The interactor will return an integer *code* (*code* $\in \{0, 1\}$) in one line. If *code* = 1, you will get `Wrong Answer` because your answer failed to match the actual locations of the rooks or you did not find enough rooks. If *code* = 0, you will enter the next test case if the current test case is not the last one.

If you attempt to print *opt* that is neither '?' nor '!', or if your binary matrix is badly formatted, such as some of the n rows has a length other than n or contains a character that is neither '0' nor '1', the interactor will return *code* = -1 and you will get `Wrong Answer` as well.

Please terminate your program immediately when you receive some *code* other than 0 to avoid any unexpected verdicts.

Example

standard input	standard output
1 3 0 110 111 111 0 111 111 101 0	 ? 101 000 101 ? 111 100 100 ! 010 001 100

Note

The blank lines in the sample case are added for readability. In your output, extra spaces or blank lines will be ignored.

Problem 167. Distinct Game

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

Given two arrays $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_m$ such that $n+m = 2k$. All values of both arrays are from 1 to k. Every value from 1 to k appears exactly twice in arrays, both occurrences could be either in the same array or in different ones.

Two players play a game. In each move, the player can take the last element of either array. The game ends when all elements are taken. The second player wins if all his elements are distinct, otherwise, the first player wins. Determine which player has a winning strategy.

Input

The first line contains a single integer t ($1 \le t \le 10^5$) — the number of test cases. The description of test cases follows.

The first line of each test case contains two integers n and m ($1 \le n, m \le 5 \cdot 10^5$) — the length of the arrays.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le \frac{n+m}{2}$) — elements of the first array.

The second line of each test case contains m integers $b_1, b_2, \ldots, b_m$ ($1 \le b_i \le \frac{n+m}{2}$) — elements of the second array.

Each value from 1 to $\frac{n+m}{2}$ appears exactly twice in the arrays.

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

Output

For each test case, print `1` if the first player wins and `2` otherwise.

Example

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