Problem A. Towers

Input file:	`standard input`
Output file:	`standard output`
Time limit (C++):	2 seconds
Time limit (Java):	4 seconds
Time limit (Python):	20 seconds
Memory limit:	256 megabytes
Java Class Name:	`towers`
Maximum Points Available:	100

Alison just got a job at a telecommunications company. The company has various cellular towers positioned in a straight line and her job is to determine the minimum signal strength these towers must output to reach all cities.

N points are given on a straight line denoting the x-coordinates of the cities. M points are also given denoting the x-coordinates of the cellular towers. All the cities and cellular towers have y-coordinate of 0. Each tower works by sending and receiving signals from cities no more than a distance r away. Your task is to determine the minimal r such that, for each city, there exists a cellular tower a distance no more than r away.

If $r = 0$ then the cellular tower only provides signal at the exact point where it is located. A signal tower may provide signal for several different cities.

Input

The first line consists of two space-separated positive integers N and M ($1 \leq N, M \leq 10^6$) where N is the number of cities and M is the number of cellular towers.

The second line consists of a sequence of N integers $a_1, a_2, \ldots, a_N$ ($-10^9 \leq a_i \leq 10^9$) denoting the coordinates of the cities. Any number of cities may be given at the same point. All coordinates a_i are given in non-decreasing order.

The third line consists of a sequence of M integers $b_1, b_2, \ldots b_M$ ($-10^9 \leq b_i \leq 10^9$) denoting the coordinates of the cellular towers. Any number of cellular towers may be given at the same point. All coordinates b_i are given in non-decreasing order.

Output

Output a single integer r denoting the minimum signal strength each tower can output to ensure that each city will be covered.

Scoring

Subtask 1 (5 points): There is only one city ($N = 1$) and $1 \leq M \leq 25$

Subtask 2 (10 points): There is only one cellular tower ($M = 1$)

Subtask 3 (15 points): $1 \leq N, M \leq 1\,000$

Subtask 4 (15 points): $1 \leq N \leq 1\,000$

Subtask 5 (15 points): $1 \leq M \leq 1\,000$

Subtask 6 (40 points): No further restrictions

Examples

standard input	standard output
3 2 -2 2 4 -3 0	4
5 3 1 5 10 14 17 4 11 15	3

Problem B. Stargazing

Input file:	`standard input`
Output file:	`standard output`
Time limit (C++):	2 seconds
Time limit (Java):	4 seconds
Time limit (Python):	20 seconds
Memory limit:	256 megabytes
Java Class Name:	`stargazing`
Maximum Points Available:	100

From a young age, Saya has always enjoyed stargazing with her friends. One day, she decides to start working on a star map so she can make her own constellations.

Saya's constellations are very simple. A star is part of a constellation if it has a Manhattan distance less than or equal to D units from any star already in that constellation. If there are no other stars within D units, this star becomes a new constellation. If there are two or more constellations within D units, these constellations combine to form one constellation.

She starts with an empty star map. On each day that she is out stargazing, she adds a new star she likes and records its position in the sky, denoted by $(X_i,\ Y_i)$. Every time she records a new star, she is curious to see what is the new size of the constellation to which that star has been added.

Over the years, she collects a total of N stars on her star map. Help her figure out the constellation size for each star at the moment they were added.

Note: The Manhattan distance is the shortest distance between two points if you could only travel up, down, left or right along the grid. In other words, for points a and b, their Manhattan distance is $|X_a - X_b| + |Y_a - Y_b|$ (where $|...|$ is the absolute value function. It essentially makes a number positive if it's negative).

Input

The first line contains two space-separated integers N and D ($0 \leq D \leq 10^6$), the total number of stars and the maximum Manhattan constellation distance (as defined above), respectively.

The next N lines contain X_i and Y_i ($0 \leq X_i, Y_i \leq 10^6$), the X and Y position of the i^{th} star Saya records.

Each star has unique coordinates.

Output

Output N numbers, each on a new line, where the i^{th} number is the constellation size of the constellation the i^{th} star was added to.

Scoring

Subtask 1 (15 points): $0 \leq X_i, Y_i, D \leq 100$, $1 \leq N \leq 100$, constellations will grow but no two constellations will ever merge.

Subtask 2 (15 points): $0 \leq X_i, Y_i, D \leq 100$, $1 \leq N \leq 100$

Subtask 3 (30 points): $1 \leq N \leq 100\,000$, $1 \leq D \leq 100$

Subtask 4 (40 points): $1 \leq N \leq 100\,000$

Example

standard input	standard output
8 5 3 5 14 8 0 3 5 11 4 8 10 15 15 8 2 7	1 1 2 1 4 1 2 5

Note

The process for each of the 8 stars is as follows:

1. When the first star is added to the empty map, there are no existing constellations, so it has to start a new one for this star. This constellation has 1 star in it, so it outputs 1.
2. This star has a distance of 14 units from the first star, and so cannot join its constellation. It instead creates a new constellation, also of size 1.
3. This star is 5 units away from the first star, and since $D = 5$ we can add it to the first star's constellation. This outputs a 2.
4. The closest other star from here is 8 units away, so this star is also part of a new constellation. It outputs a 1.
5. This star is an interesting case. It is 4 units away from the previous star (constellation size 1), and also 4 units away from the first star (part of a constellation of size 2). This star joins these two existing constellations together, producing a constellation with a total size of 4 stars.
6. This star is on its own.
7. This star joins the second star's constellation.
8. The last star is close to the first and 5th star, but both of these are already in the same constellation. It joins this constellation as well, bringing the total size to 5 stars.

Problem C. Owls

Input file:	`standard input`
Output file:	`standard output`
Time limit (C++):	5 seconds
Time limit (Java):	10 seconds
Time limit (Python):	50 seconds
Memory limit:	256 megabytes
Java Class Name:	`owls`
Maximum Points Available:	100

A parliament of K owls are organising a meeting, and need help with the perching arrangements. They are meeting in a forest where there are N convenient perches. The owls want to be sure that they can all see each other. Owls can turn their heads right around, so the only reason two owls might not be able to see each other is if there is another owl directly between them.

You've been asked to come up with a list of valid perch arrangements, but have realised that there might be a lot of them. Count the number of ways to assign the K owls to distinct perches such that no owl is directly between two other owls. Two assignments are considered different if there is some perch that is occupied in one assignment but not the other. In particular, swapping owls between perches does not count as a new assignment. Since the answer may be very large, return the count modulo $10^9 + 7$ (that is, the remainder when divided by $10^9 + 7$).

Input

The first line of input contains two integers, N and K ($3 \leq K \leq N \leq 1000$). The following N lines each contain two integers x and y ($0 \leq x, y \leq 10\,000$), giving the coordinates of one of the perches. No two perches have the same position.

Output

Output a single integer, the number of valid perch assignments, modulo $10^9 + 7$.

Scoring

Subtask 1 (15 points): $K \leq 5$, $N \leq 30$

Subtask 2 (20 points): $K = 3$

Subtask 3 (25 points): $K = 4$

Subtask 4 (40 points): $K = 5$

Example

standard input	standard output
7 4 0 0 0 5 0 9 5 5 10 5 5 7 10 14	19