Programming Assignments

Computational Geometry

October 17, 2019

Welcome to the "Computational Geometry" MOOC at Coursera! This document contains the statements of all programming assignments of the course. There are five programming assignments (one for each week), each containing four problems. To pass a programming assignment, you need to solve at least two problems from it. For the first two problems in each assignment, we provide starter solutions in `C++`, `Java`, and `Python`.

When you pass a programming assignment (i.e., when you successfully solve any two of its problems), you will get access to a locked reading that contains solution to the fourth problem of this programming assignment. We strongly encourage you to take a look at these solutions *only after* you've tried to solve this problem yourself: it will be more useful for you if you solve the problem without reading its solution (not to mention satisfaction and self-confidence that you get upon solving :)). If you decide to read the solution, make sure to implement it and to submit to the grading system. This will allow you to understand it even better, and to improve your implementation skills.

You may submit solutions in any of the ten supported programming languages listed in the end of this document. We believe that the time/memory limits allow to solve each of the problems in all of these languages, but we have only checked this for `C++`, `Java`, and `Python` (hence, we provide no guarantees for other languages, use them at your own risk).

The time limit "n seconds" in a problem statement means the following: n seconds for `C`, `C++`, `Rust`; $1.5n$ seconds for `Java` and `C#`; $3n$ seconds for `Scala`; $5n$ seconds for `JavaScript`, `Python`, `Ruby`.

Contents

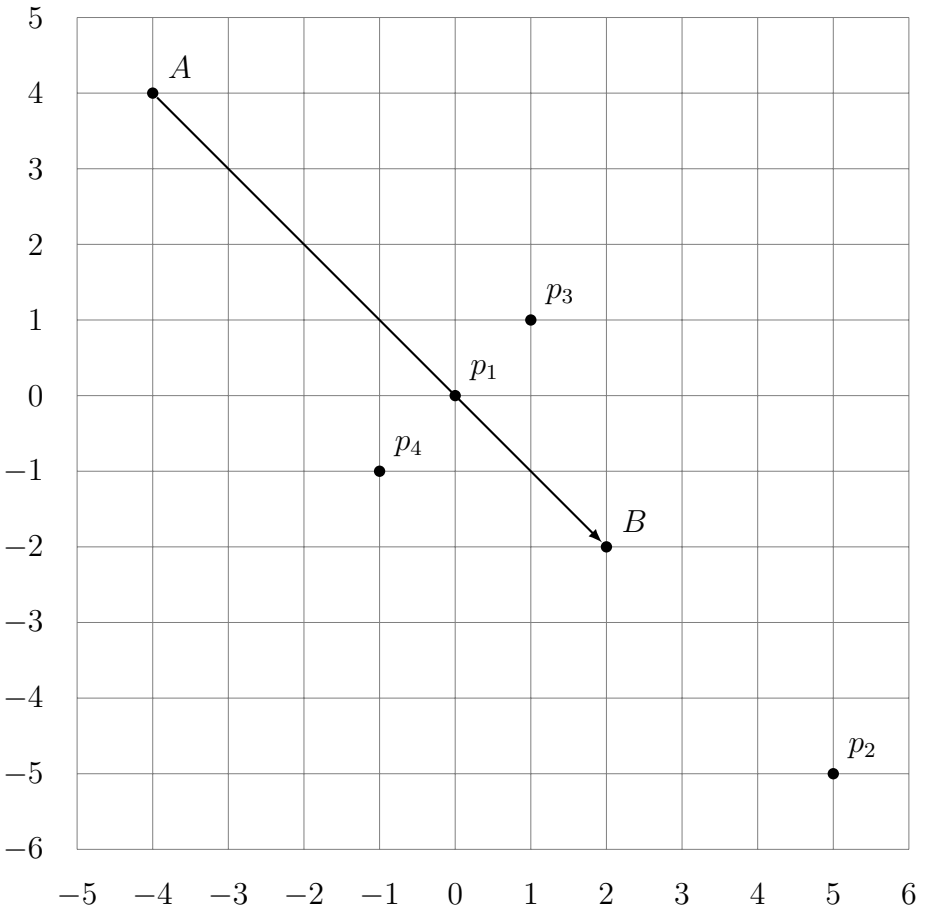
1 Week 1: Point inclusion in a polygon

1.1 Points and Vector

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.25 seconds
Memory limit:	256 megabytes

Given a vector AB (starting at point A and ending at point B) and n points $p_1, p_2, \ldots, p_n$ on the plane, determine the relative position of each point with respect to the vector: point lies either on the right side, or on the left side, or on the vector, or on the continuation of the vector.

For example, if $A = (-4,\ 4)$ and $B = (2,\ -2)$ are the beginning and end of the vector, respectively, then the point $p_1 = (0, 0)$ is on the vector, the point $p_2 = (5,\ -5)$ is on the continuation of the vector, the point $p_3 = (1,\ 1)$ is on the left side, while the point $p_4 = (-1,\ -1)$ on the right side.



Input

The first line contains four integers A_x A_y B_x B_y, the coordinates of the beginning and end of the vector.

The second line contains an integer n ($1 \le n \le 1\,000$), the number of points. The i^{th} of the next n lines specifies a point p_i by two integer coordinates $p_{i,x}$ and $p_{i,y}$.

It is guaranteed that each coordinate does not exceed 10^4 by absolute value and that the vector is not degenerate (i.e., has non-zero length).

Output

For each point p_i, output one of the following:

- `LEFT`, if point p_i is to the left of the vector;
- `RIGHT`, if point p_i is to the right of the vector;
- `ON_SEGMENT`, if point p_i is on the vector;
- `ON_LINE`, if point p_i is on the continuation of the vector.

Examples

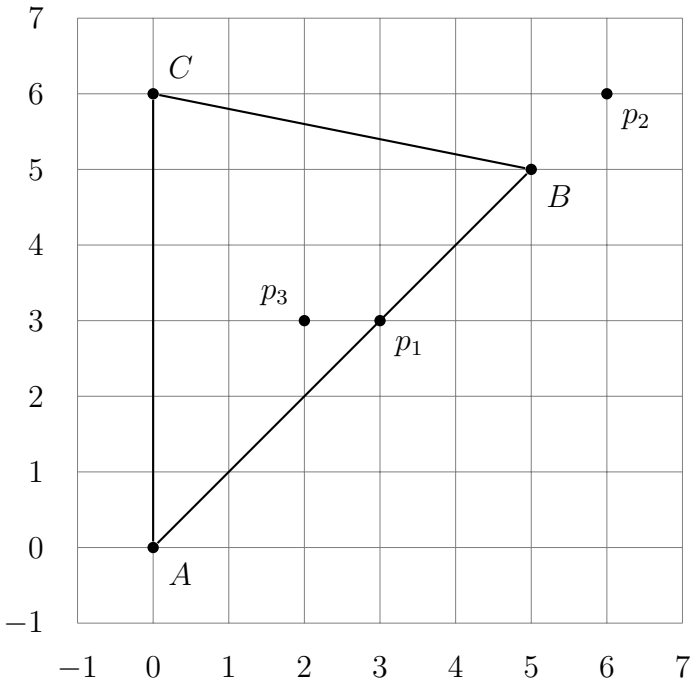
standard input	standard output
-4 4 2 -2 4 0 0 5 -5 1 1 -1 -1	ON_SEGMENT ON_LINE LEFT RIGHT
-4 0 2 -10 8 -8 4 2 -7 9 3 -4 6 -8 -5 7 1 2 4 -3 -9	RIGHT LEFT LEFT LEFT RIGHT LEFT LEFT RIGHT

1.2 Points and Triangle

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.25 seconds
Memory limit:	256 megabytes

Given a triangle ABC and n points $p_1, p_2, \ldots, p_n$ on the plane, determine for each point whether it is inside, outside, or on the border of the triangle.

For example, for $A = (0, 0)$, $B = (5, 5)$, $C = (0, 6)$, a point $p_1 = (3, 3)$ lies on the border, $p_2 = (6, 6)$ lies outside, and $p_3 = (2, 3)$ lies inside the triangle.



Input

The first line contains six integers A_x, A_y, B_x, B_y, C_x, C_y, the coordinates of triangle vertices.

The second line contains an integer n ($1 \le n \le 1\,000$), the number of points. The i-th of the following n lines contains two integers $p_{i,x}$ $p_{i,y}$, the coordinates of the i-th point p_i.

Each coordinate does not exceed 10^4 in the absolute value. It is guaranteed that the triangle is not degenerate.

Output

For each point p_i, print on a separate line one of the following three messages:

- `INSIDE`, if point p_i is strictly inside the triangle.
- `OUTSIDE`, if point p_i is outside the triangle.
- `BORDER`, if point p_i is on the border of the triangle.

Examples

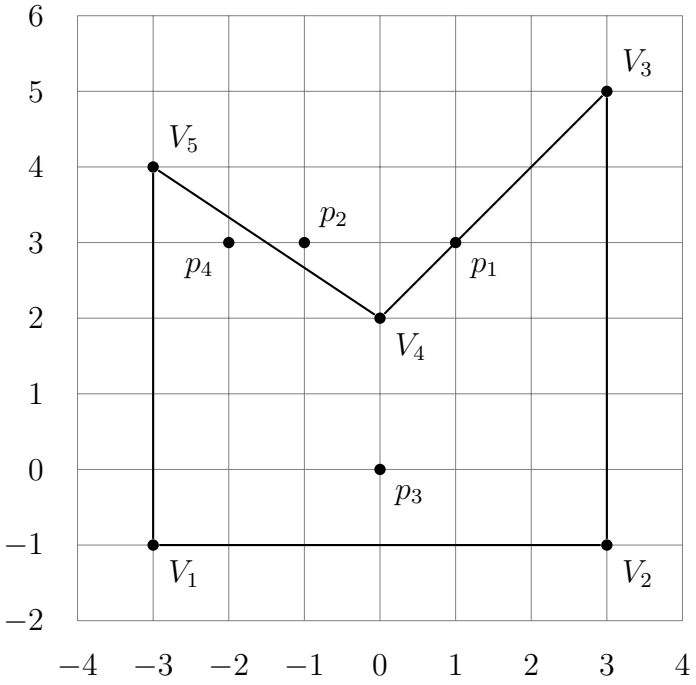
standard input	standard output
0 0 5 5 0 6 3 3 3 6 6 2 3	BORDER OUTSIDE INSIDE
1 0 0 3 3 3 12 0 0 0 1 0 2 0 3 1 0 1 1 1 2 1 3 2 0 2 1 2 2 2 3	OUTSIDE OUTSIDE OUTSIDE BORDER BORDER INSIDE INSIDE BORDER OUTSIDE OUTSIDE INSIDE BORDER

1.3 Points and Polygon

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

Given a simple polygon $V_1V_2 \cdots V_m$ and n points $p_1, p_2, \ldots, p_n$ on the plane, determine for each point whether it is inside, outside, or on the border of the polygon.

For example, if points $V_1 = (-3, -1)$, $V_2 = (3, -1)$, $V_3 = (3, 5)$, $V_4 = (0, 2)$, $V_5 = (-3, 4)$ are the vertices of the polygon, then $p_1 = (1, 3)$ is located on the border, $p_2 = (-1, 3)$ is outside the polygon, and $p_3 = (0, 0)$ and $p_4 = (-2, 3)$ are inside the polygon.



Input

The first line contains an integer m ($3 \leq m \leq 200$), the number of vertices of the polygon.

The second line contains $2m$ integers $V_{1,x}, V_{1,y}, \ldots, V_{m,x}, V_{m,y}$, the coordinates of the polygon vertices. The vertices listed in counterclockwise order.

The third line contains an integer n ($1 \leq n \leq 200$), the number of points. The i-th of the following n lines contains two integers $p_{i,x}$ and $p_{i,y}$.

It is guaranteed that each coordinate does not exceed 10^4 in the absolute value, no three consecutive vertices lie on the same line, and there is neither self-crossing nor self-touching of the polygon.

Output

For each point p_i, output one of the following:

- `INSIDE`, if point p_i is strictly inside the polygon.
- `OUTSIDE`, if point p_i is outside the polygon.
- `BORDER`, if point p_i is on the border of the polygon.

Example

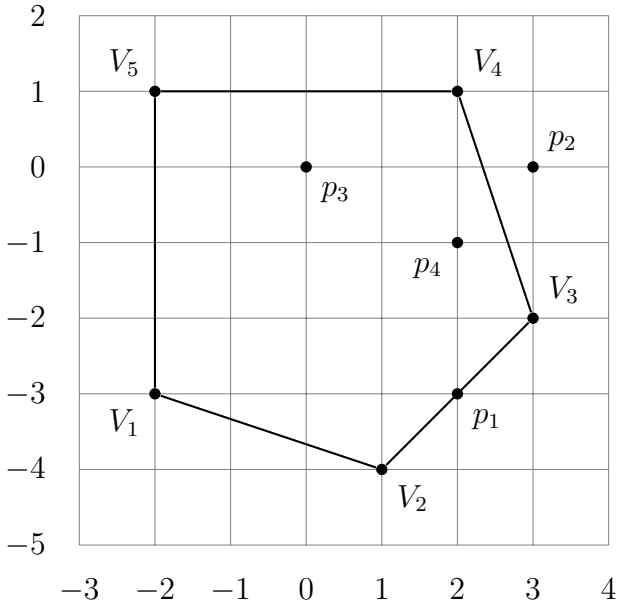
standard input	standard output
5 -3 -1 3 -1 3 5 0 2 -3 4 4 1 3 -1 3 0 0 -2 3	BORDER OUTSIDE INSIDE INSIDE

1.4 Points and Convex Polygon

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

Given a simple *convex* polygon $V_1V_2 \cdots V_m$ and n points $p_1, p_2, \ldots, p_n$ on the plane, determine which of them are located inside, outside or on the border of the polygon.

For example, if points $V_1 = (-2, -3)$, $V_2 = (1, -4)$, $V_3 = (3, -2)$, $V_4 = (2, 1)$, $V_5 = (-2, 1)$ are the vertices of the polygon, then $p_1 = (2, -3)$ is located on the border, $p_2 = (3, 0)$ is outside the polygon, and $p_3 = (0, 0)$, $p_4 = (2, -1)$ are inside the polygon.



Input

The first line contains an integer m ($3 \leq m \leq 20\,000$), the number of vertices.

The second line contains $2m$ integers $V_{1,x}, V_{1,y}, \ldots, V_{m,x}, V_{m,y}$, the coordinates of polygon vertices. Vertices listed in the counterclockwise order.

The third line contains an integer n ($1 \leq n \leq 50\,000$), the number of points. The i-th of the following n lines contains two integers $p_{i,x}$ $p_{i,y}$.

It is guaranteed that each coordinate does not exceed 10^6 in the absolute value and no three consecutive vertices lie on the same line.

Output

For each point p_i, print one of the following:

- `INSIDE`, if point p_i is strictly inside the polygon.
- `OUTSIDE`, if point p_i is outside the polygon.
- `BORDER`, if point p_i is on the border of the polygon.

Example

standard input	standard output
5 -2 -3 1 -4 3 -2 2 1 -2 1 4 2 -3 3 0 0 0 2 -1	BORDER OUTSIDE INSIDE INSIDE

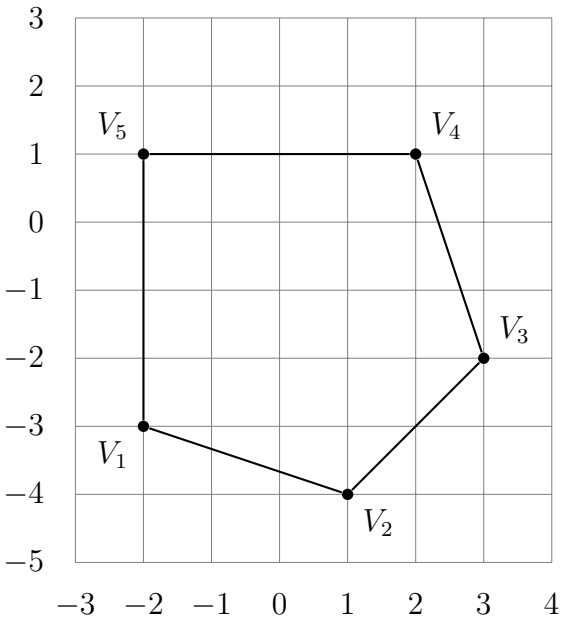
2 Week 2: Convex hulls

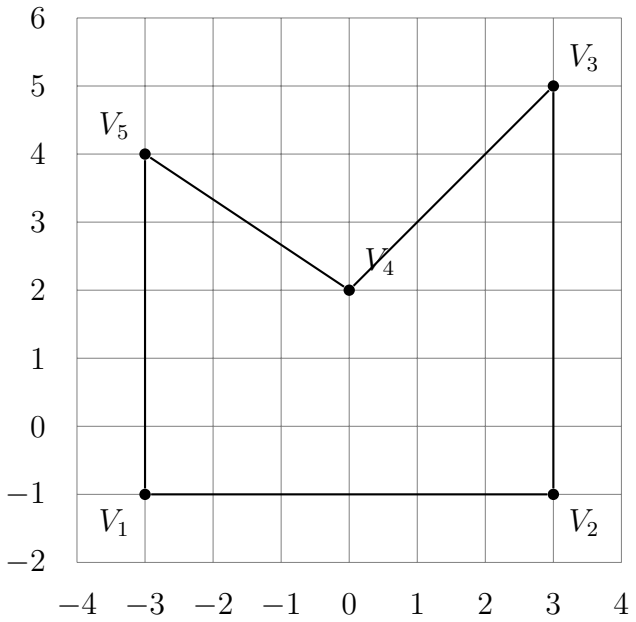
2.1 Convex polygon

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.25 seconds
Memory limit:	256 megabytes

Given a simple polygon $V_1\ V_2\ \ldots\ V_n$, determine whether it is convex.

For example, the polygon with vertex list $(-2,\ -3)$, $(1,\ -4)$, $(3,\ -2)$, $(2,\ 1)$, $(-2,\ 1)$ is convex, while the polygon with vertex list $(-3,\ -1)$, $(3,\ -1)$, $(3,\ 5)$, $(0,\ 2)$, $(-3,\ 4)$ is not convex.





Input

The first line contains an integer n ($3 \le n \le 500$), the number of vertices.

The second line contains $2n$ integers $V_{1,x}$ $V_{1,y}$ $V_{2,x}$ $V_{2,y}$ ... $V_{n,x}$ $V_{n,y}$, the coordinates of polygon vertices. The vertices listed in the counterclockwise order.

It is guaranteed that each coordinate does not exceed 10^4 by absolute value and that no three consecutive vertices lie on the same line.

Output

Output one message:

- `CONVEX`, if polygon is convex.
- `NOT_CONVEX`, if polygon is not convex.

Examples

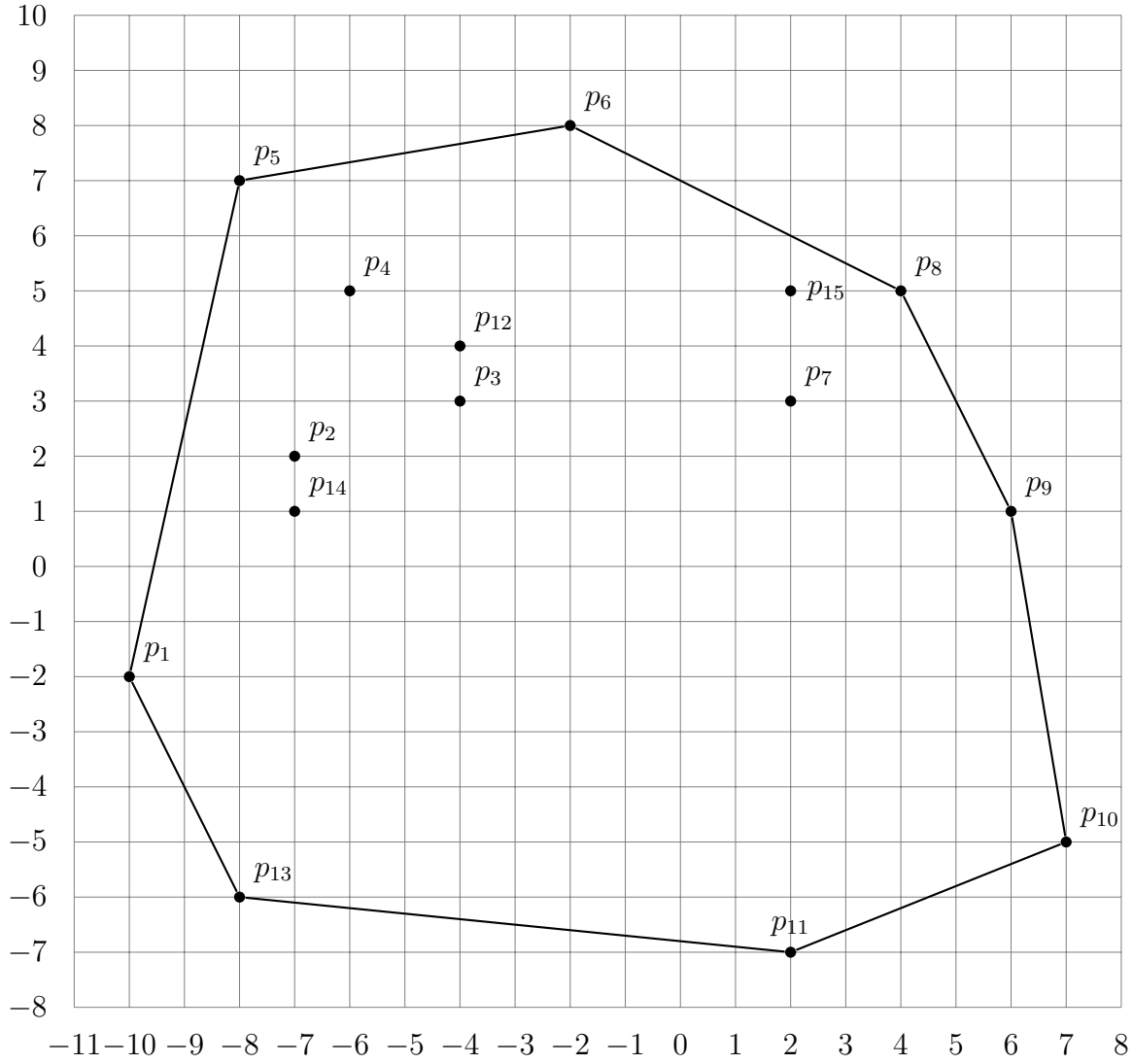
standard input	standard output
5 -3 -1 3 -1 3 5 0 2 -3 4	NOT_CONVEX
5 -2 -3 1 -4 3 -2 2 1 -2 1	CONVEX

2.2 Build convex hull

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.5 seconds
Memory limit:	256 megabytes

Build a convex hull of the given n points (i.e., build a convex polygon with a minimum perimeter that contains all the points).

For example, the minimal convex hull of the points $p_1(-10, 2)$, $p_2(-7, 2)$, $p_3(-4, 3)$, $p_4(-6, 5)$, $p_5(-8, 7)$, $p_6(-2, 8)$, $p_7(2, 3)$, $p_8(4, 5)$, $p_9(6, 1)$, $p_{10}(7, -5)$, $p_{11}(2, -7)$, $p_{12}(-4, -4)$, $p_{13}(-8, -6)$, $p_{14}(-7, -1)$, $p_{15}(2, 6)$ has vertices $v_1(-10, 2)$, $v_2(-8, -6)$, $v_3(2, -7)$, $v_4(7, -5)$, $v_5(6, 1)$, $v_6(4, 5)$, $v_7(-2, 8)$, $v_8(-8, 7)$.



Input

The first line contains an integer n ($3 \le n \le 5000$), the number of points.

The second line contains $2n$ integers $p_{1,x}\ p_{1,y}\ p_{2,x}\ p_{2,y} \ldots p_{n,x}\ p_{n,y}$, the coordinates of points.

It is guaranteed that each coordinate does not exceed 10^6 by absolute value and that there are no three points lying on the same line.

Output

In the the first line, output an integer m, the number of vertices of the resulting minimal convex hull.

In the second line, output $2m$ integers $p_{1,x}\ p_{1,y}\ p_{2,x}\ p_{2,y} \ldots p_{m,x}\ p_{m,y}$, the coordinates of the vertices of the convex hull, in the counterclockwise order.

Examples

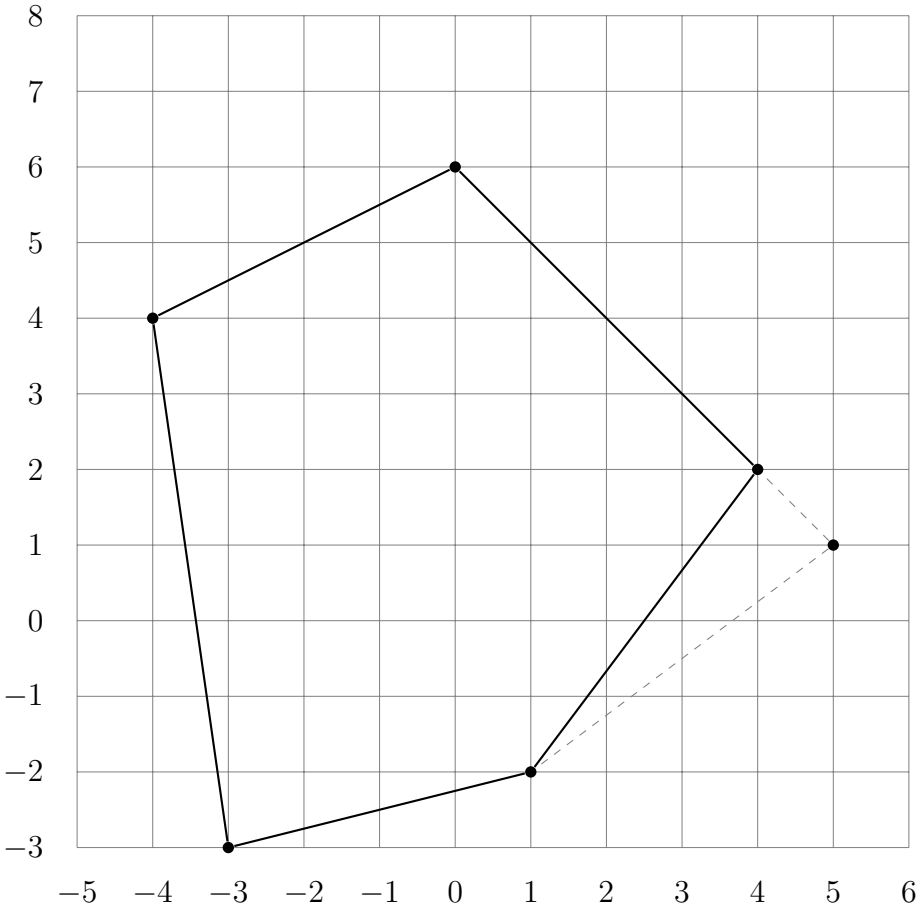
standard input
15 -10 2 -7 2 -4 3 -6 5 -8 7 -2 8 2 3 4 5 6 1 7 -5 2 -7 -4 -4 -8 -6 -7 -1 2 6
standard output
8 -8 -6 2 -7 7 -5 6 1 4 5 -2 8 -8 7 -10 2
standard input
10 -3 11 -7 10 -9 7 -10 0 -11 -12 4 -6 6 -5 7 4 6 7 4 9
standard output
10 4 -6 6 -5 7 4 6 7 4 9 -3 11 -7 10 -9 7 -10 0 -11 -12
standard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

2.3 Tangents to polygon

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

Given a simple *convex* polygon $V_1\ V_2\ \ldots\ V_m$ and n points $p_1,\ p_2, \ldots,\ p_n$ on the plane, for each point, build two tangents to the polygon. If the tangent lies on the continuation of the side of the polygon, output any vertex from this side.

For example, for a polygon with vertex list $(-3,\ -3)$, $(1,\ -2)$, $(4,\ 2)$, $(0,\ 6)$, $(-4,\ 4)$, the tangents drawn through the point $(5,\ 1)$ pass through the vertices $v_l(1,\ -2)$, $v_r(0,\ 6)$.



Input

The first line contains an integer m ($3 \leq m \leq 1000$), the number of vertices.

The second line contains $2m$ integers $V_{1,x}$ $V_{1,y}$ $V_{2,x}$ $V_{2,y} \ldots$ $V_{m,x}$ $V_{m,y}$, the coordinates of polygon vertices. The vertices listed in counterclockwise order.

The third line contains an integer n ($1 \leq n \leq 1000$), the number of points.

The i^{th} of the next n lines contains two integers $p_{i,x}$ $p_{i,y}$.

It is guaranteed that each coordinate does not exceed 10^6 by absolute value and that each point is outside of the polygon.

Output

For each point p_i, output four integers specifying two vertices V_{lx}, V_{ly}, V_{rx}, V_{ry} through which left and right tangents pass, respectively.

Examples

standard input
4 23 16 -21 -12 -7 -24 18 -25 5 -7 12 24 0 -12 -5 -13 -25 -9 16
standard output
23 16 -21 -12 18 -25 23 16 23 16 -21 -12 -21 -12 18 -25 23 16 -21 -12

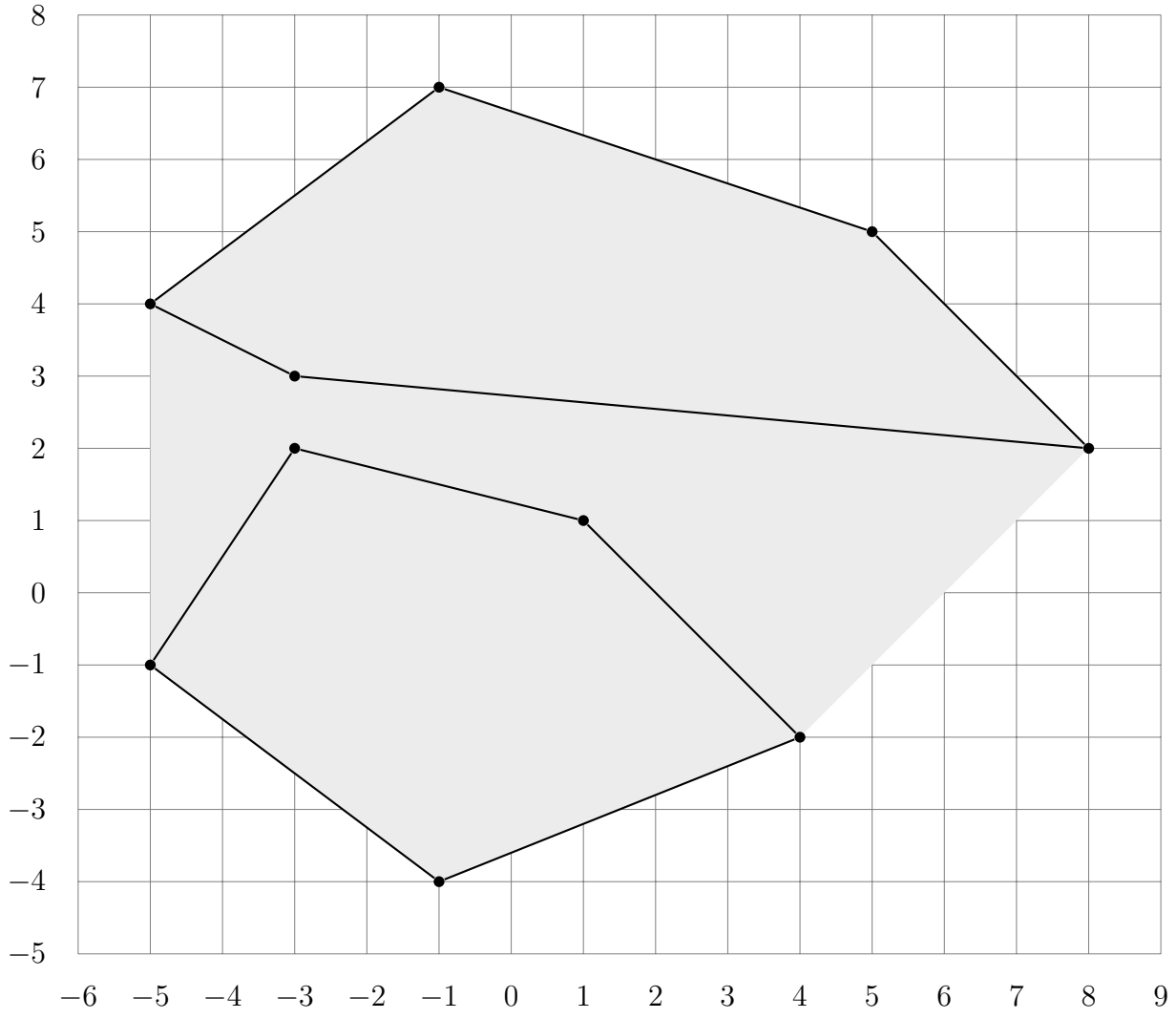
standard input
10 -19 20 -23 17 -22 -16 -10 -18 -4 -17 2 -12 12 1 13 3 8 14 2 19 5 -16 20 17 7 -10 -23 -24 8 19 -18
standard output
2 19 -19 20 2 -12 2 19 -22 -16 2 -12 -23 17 -22 -16 -10 -18 13 3

2.4 Union of convex polygons

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

Compute the union of the given convex polygons.

For example, if the first polygon is given by vertices $(-5,\ 4)$, $(-3,\ 3)$, $(2,\ 2)$, $(8,\ 2)$, $(5,\ 5)$, $(-1,\ 7)$, and the second is given by vertices $(-5,\ -1)$, $(-1,\ -4)$, $(4,\ -2)$, $(1,\ 1)$, $(-3,\ 2)$, then their union is a convex polygon with vertices: $(-5,\ 4)$, $(-5,\ -1)$, $(-1,\ -4)$, $(4,\ -2)$, $(8,\ 2)$, $(5,\ 5)$, $(-1,\ 7)$.



Input

The first line contains an integer n ($n \geq 2$), the number of polygons.

The next $2n$ lines contain the description of polygons in the following format. The i^{th} line contains an integer m ($m \geq 3$), the number of vertices. The $(i+1)^{th}$ line contains $2m$ integers $V_{1,x}$ $V_{1,y}$ $V_{2,x}$ $V_{2,y} \ldots V_{m,x}$ $V_{m,y}$, the coordinates of polygon vertices. The vertices are listed in counterclockwise order.

It is guaranteed that each coordinate does not exceed 10^6 by absolute value, that no three consecutive vertices lie on the same line, and that the total number of vertices is less than 10^5.

Output

In the first line, output an integer m, the number of vertices in the resulting polygon. In the second line, output $2m$ integers $p_{1,x}$ $p_{1,y}$ $p_{2,x}$ $p_{2,y} \ldots p_{m,x}$ $p_{m,y}$, the coordinates of the resulting polygon. As usual, coordinates are expected in counterclockwise order.

Examples

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

standard input
3 6 2 17 -17 6 -15 -5 -6 -10 1 -11 18 15 3 -16 10 -6 -8 0 2 8 16 13 11 12 6 10 -10 -4 -14 -11 -3 -16 8 -20 15 -10
standard output
8 -14 -11 -3 -16 8 -20 15 -10 18 15 2 17 -16 10 -17 6

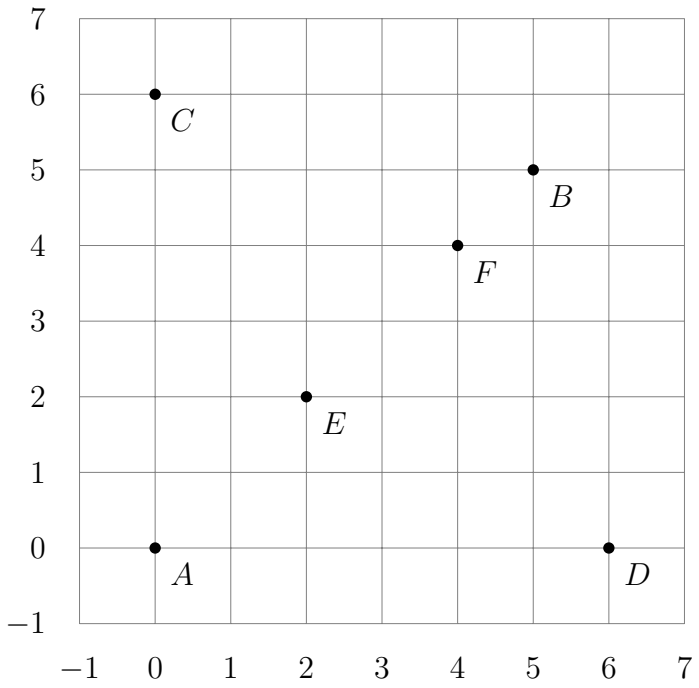
3 Week 3: Intersections

3.1 Segments Intersection

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.25 seconds
Memory limit:	256 megabytes

Given two segments, determine do they intersect or not. There are three possible outcomes:

- they do not have common points (like segments AC and BD below);
- they share a single intersection point (like EB and CD);
- they share a common segment of non-zero length (like AF and EB).



Input

The first line contains four integers A_x, A_y, B_x, B_y, the coordinates of end points of the first segment. The second line contains four integers C_x, C_y, D_x, D_y, the coordinates of end points of the second segment. Each coordinate does not exceed 10^4 in the absolute value and each segment has non-zero length.

Output

Output a single message depending on the relative position of segments (make sure to strictly follow the output format):

- `No common points.`
- `The intersection point is (`E_x`, `E_y`).`
- `A common segment of non-zero length.`

It is guaranteed that if the two input segments share a single common point E, then E_x and E_y are integers.

Examples

standard input
0 0 5 5 0 6 6 0
standard output
The intersection point is (3, 3).

standard input
0 0 5 5 5 0 3 2
standard output
No common points.

standard input
0 0 10 0 1 0 2 0
standard output
A common segment of non-zero length.

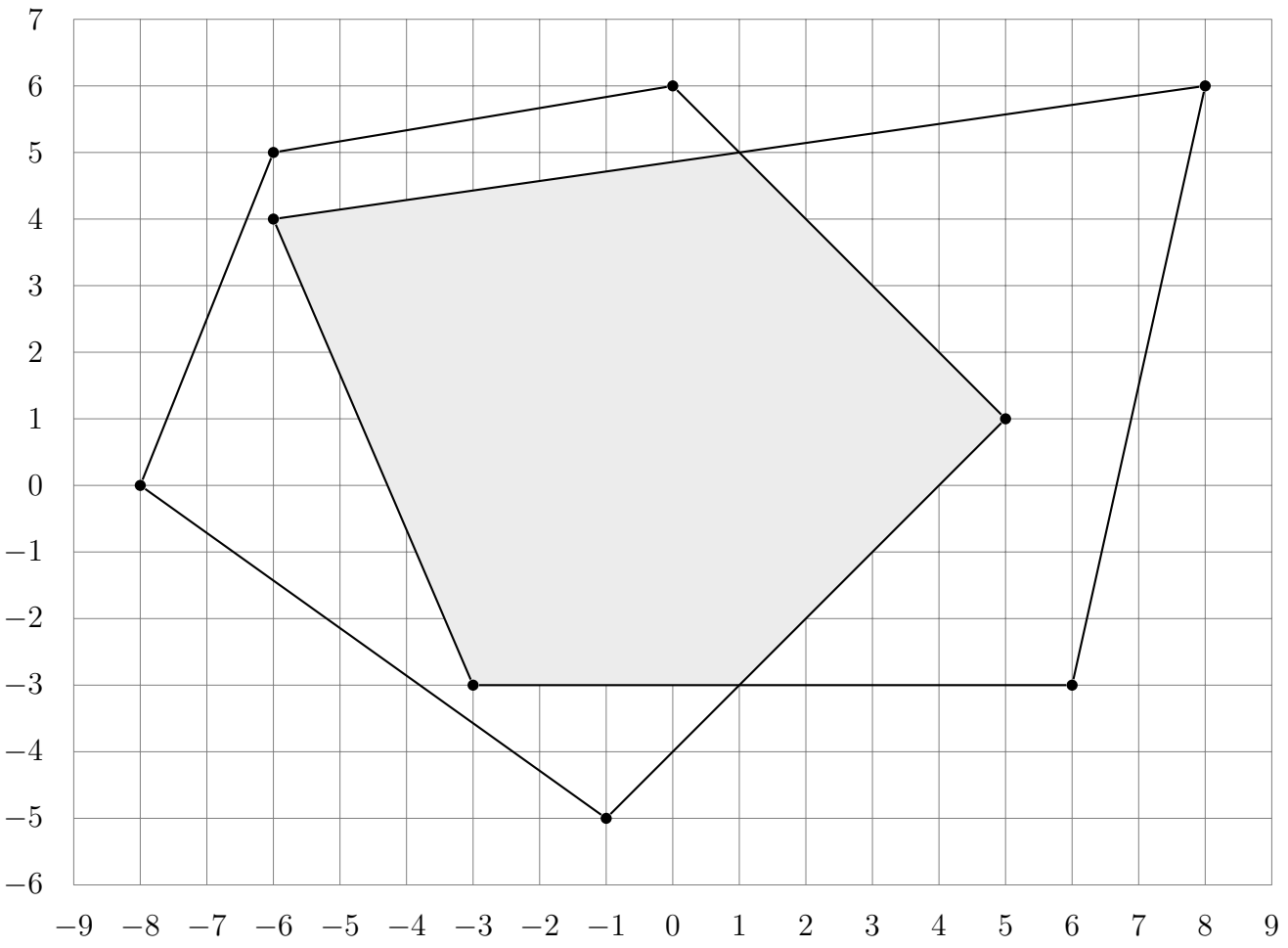
standard input
0 0 10 0 5 0 15 0
standard output
A common segment of non-zero length.

3.2 Polygon intersection

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

Find the intersection of two convex polygons.

For example, the intersection of a polygon with vertices $(-8,\ 0)$, $(-1,\ -5)$, $(5,\ 1)$, $(0,\ 6)$, $(-6,\ 5)$ and a polygon with vertices $(-6,\ 4)$, $(-3,\ -3)$, $(6,\ -3)$, $(8,\ 6)$ is a polygon with vertices $(-6,\ 4)$, $(-3,\ -3)$, $(1,\ -3)$, $(5,\ 1)$, $(1,\ 5)$.



Input

The first line contains an integer n ($3 \le n \le 100$), the number of vertices in the first polygon. The second line contains $2n$ integers $V_{1,x}\ V_{1,y}\ V_{2,x}\ V_{2,y}\ \ldots\ V_{n,x}\ V_{n,y}$, the coordinates of polygon vertices. The vertices are listed in counterclockwise order.

The third line contains an integer m ($3 \le m \le 100$), the number of vertices in the second polygon. The fourth line contains $2m$ integers $W_{1,x}\ W_{1,y}\ W_{2,x}\ W_{2,y}\ \ldots\ W_{m,x}\ W_{m,y}$, the coordinates of polygon vertices. The vertices are listed in counterclockwise order.

It is guaranteed that each coordinate does not exceed 10^3 by absolute value and that no three vertices of the same polygon lie on the same line. The coordinates of the vertices of the intersection of polygons are integers.

Output

In the first line, output an integer m, the number of vertices in the resulting polygon. In the second line, output $2m$ integers $p_{1,x}\ p_{1,y}\ p_{2,x}\ p_{2,y}\ \ldots\ p_{m,x}\ p_{m,y}$, the coordinates of the resulting polygon. Coordinates are expected in counterclockwise order. If the polygons have an empty intersection, output just 0.

Examples

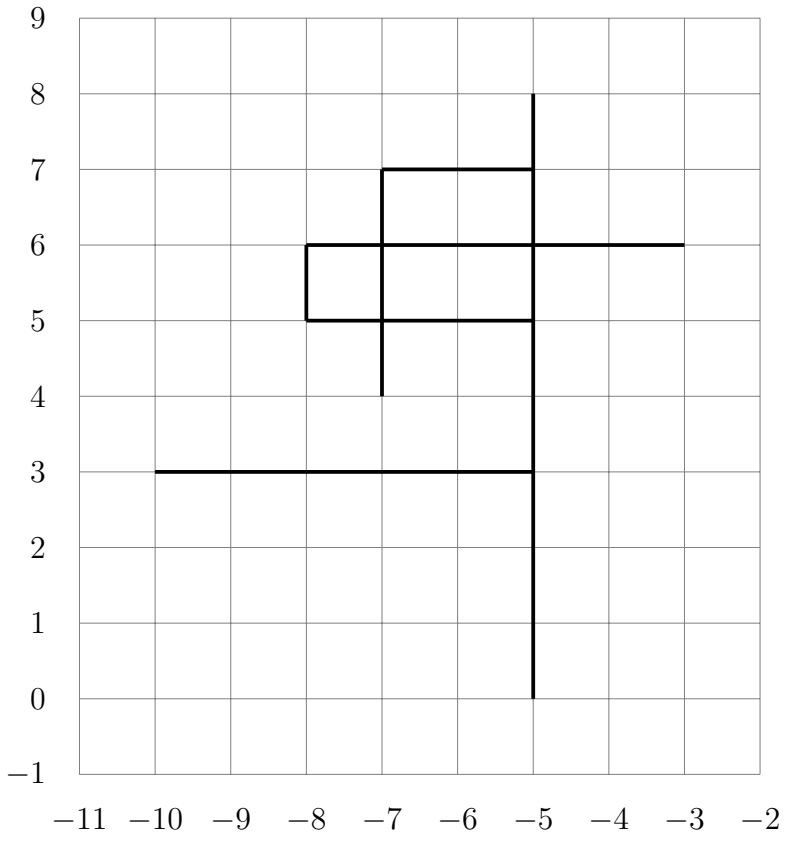
standard input	standard output
3 1 11 -8 -3 -9 -13 5 5 14 -1 0 4 -14 8 -8 12 4	0
5 -8 0 -1 -5 5 1 0 6 -6 5 4 -6 4 -3 -3 6 -3 8 6	5 -3 -3 1 -3 5 1 1 5 -6 4
4 0 0 10 0 10 10 0 10 4 5 -1 11 5 5 11 -1 5	8 10 4 10 6 6 10 4 10 0 6 0 4 4 0 6 0

3.3 The intersection of horizontal and vertical segments

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

Given n segments parallel to the coordinate axes, count the number of intersection points. No vertical segment intersects another vertical segment, no horizontal segment intersects another horizontal segment.

For example, the following seven segments (from the sample 1 below) have nine intersection points.



Input

The first line contains an integer n ($1 \leq n \leq 50\,000$), the number of segments.

The i^{th} of the next n lines contains four integers $p_{1,x}$ $p_{1,y}$ $p_{2,x}$ $p_{2,y}$ coordinates of points defining a segment.

It is guaranteed that each coordinate does not exceed 10^6 by absolute value.

Output

Output the number of intersection points.

Examples

standard input	standard output
7 -10 3 -5 3 -5 0 -5 8 -7 4 -7 7 -7 7 -5 7 -8 5 -8 6 -8 5 -5 5 -8 6 -3 6	9
8 22 -4 22 -12 -14 1 -5 1 -23 -3 -6 -3 -6 13 -6 12 -1 19 1 19 -17 4 12 4 -15 15 -15 11 -6 -22 2 -22	0

Note

Think about an appropriate data structure for solving this problem.

3.4 Intersection of a set of segments

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

Given n segments on the plane, find all pairs of intersecting segments. It is guaranteed that the number of pairs of intersecting segments is at most 5 000. All segments satisfy the following:

- There are no vertical segments.
- Endpoints of line segments do not serve as intersection points.
- No three (or more) segments have a common intersection.
- No two segments overlap.

Input

The first line contains an integer n ($1 \le n \le 30\,000$), the number of segments.

The i-th of the next n lines contains four integers $p_{1,x}$, $p_{1,y}$, $p_{2,x}$, $p_{2,y}$, the coordinates of endpoints of the i-th segment.

It is guaranteed that each coordinate does not exceed 10^5 by the absolute value.

Output

In the first line, output the number m of pairs of intersecting segments.

In each of the following m lines, output eight integers: the first four integers are endpoints of the first segment, the next four integers are endpoints of the second segment.

Examples

standard input	standard output
3 -7 -3 2 6 -7 2 9 -2 3 -3 -3 6	3 -7 2 9 -2 -7 -3 2 6 -3 6 3 -3 -7 -3 2 6 -3 6 3 -3 -7 2 9 -2
9 29 48 30 13 49 57 -68 -24 17 -30 55 16 19 -61 91 -2 44 -22 21 73 -68 -38 -77 90 -74 84 55 -32 26 72 -91 64 -21 -41 -3 75	13 -77 90 -68 -38 -91 64 26 72 -74 84 55 -32 -91 64 26 72 -68 -24 49 57 -21 -41 -3 75 -74 84 55 -32 -21 -41 -3 75 -91 64 26 72 -21 -41 -3 75 -74 84 55 -32 -68 -24 49 57 21 73 44 -22 -91 64 26 72 21 73 44 -22 -68 -24 49 57 29 48 30 13 -68 -24 49 57 29 48 30 13 21 73 44 -22 -74 84 55 -32 17 -30 55 16 21 73 44 -22 17 -30 55 16 -74 84 55 -32 19 -61 91 -2

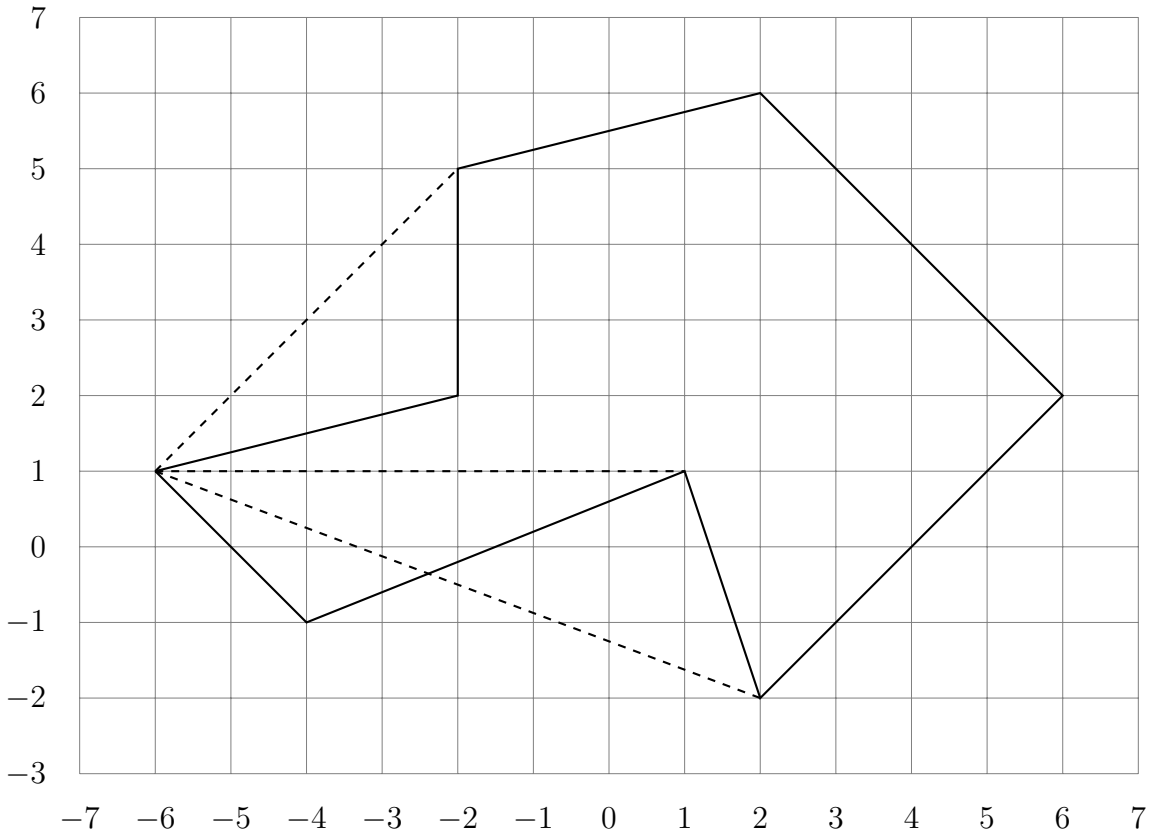
4 Week 4: Polygon triangulation

4.1 Diagonal

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

Given a simple polygon, check, for each of its diagonals, whether it is external, internal, or intersects the side.

For example, for a polygon with vertex set $(-6, 1), (-4, -1), (1, 1), (2, -2), (6, 2), (2, 6), (-2, 5), (-2, 2)$, the diagonal $(-6, 1)$—$(1, 1)$ is internal, the diagonal $(-6, 1)$—$(2, 5)$ is external, and the diagonal $(-6, 1)$—$(2, -2)$ is intersecting.



Input

The first line contains an integer n ($4 \le n \le 50$), the number of vertices.

The second line contains $2n$ integers $V_{1,x}, V_{1,y}, V_{2,x}, V_{2,y}, \ldots, V_{n,x}, V_{n,y}$, the coordinates of polygon vertices. The vertices are listed in counterclockwise order. It is guaranteed that each coordinate does not exceed 10^4 by the absolute value.

Output

In the first line, output the number m of diagonals.

In each of the next m lines, output the coordinates of two endpoint of a diagonal (in any order) and its type:

- `INTERSECT`, if the diagonal crosses the side;
- `INTERNAL`, if the diagonal is internal;
- `EXTERNAL`, if the diagonal is external.

Example

standard input
8 -6 1 -4 -1 1 1 2 -2 6 2 2 6 -2 5 -2 2
standard output
20 -6 1 1 1 INTERNAL -6 1 2 -2 INTERSECT -6 1 6 2 INTERNAL -6 1 2 6 INTERSECT -6 1 -2 5 EXTERNAL -4 -1 2 -2 EXTERNAL -4 -1 6 2 INTERSECT -4 -1 2 6 INTERNAL -4 -1 -2 5 INTERSECT -4 -1 -2 2 INTERNAL 1 1 6 2 INTERNAL 1 1 2 6 INTERNAL 1 1 -2 5 INTERNAL 1 1 -2 2 INTERNAL 2 -2 2 6 INTERNAL 2 -2 -2 5 INTERSECT 2 -2 -2 2 INTERSECT 6 2 -2 5 INTERNAL 6 2 -2 2 INTERNAL 2 6 -2 2 INTERNAL

4.2 Triangulation of polygon by ear-cutting algorithm

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

Triangulate a given polygon by the ear-cutting algorithm.

Input

The first line contains an integer n ($4 \le n \le 100$), the number of vertices.

The second line contains $2n$ integers $V_{1,x}, V_{1,y}, V_{2,x}, V_{2,y}, \ldots, V_{n,x}, V_{n,y}$, the coordinates of polygon vertices. The vertices are listed in counterclockwise order. It is guaranteed that each coordinate does not exceed 10^4 by the absolute value.

Output

In the first line, output the number m of triangles. In each of the next m lines, output the coordinates of the cut triangle. The triangles are expected in the order they are cut by the ear-cutting algorithm. (Therefore, the last one is the remaining triangle.)

Example

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

4.3 Monotone polygon triangulation

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

Triangulate a given vertically strictly monotonous polygon. (Recall that a polygon is called vertically strictly monotone if any horizontal line intersects its contour in at most two points.)

Input

The first line contains an integer n ($4 \le n \le 5\,000$), the number of vertices.

The second line contains $2n$ integers $V_{1,x}, V_{1,y}, V_{2,x}, V_{2,y}, \ldots, V_{n,x}, V_{n,y}$, the coordinates of polygon vertices. The vertices are listed in counterclockwise order. It is guaranteed that each coordinate does not exceed 10^5 by the absolute value.

Output

In the first line, output the number m of diagonals carried out during the triangulation process. In each of the following m lines, output the coordinates of the endpoints of the clip diagonal.

Examples

standard input
6 0 8 -5 6 -2 0 1 -8 5 -1 11 7
standard output
3 -5 6 11 7 -2 0 11 7 5 -1 -2 0
standard input
8 -17 12 -6 5 0 -7 -30 -13 3 -14 21 -12 34 -10 26 11
standard output
5 -6 5 26 11 0 -7 26 11 34 -10 0 -7 21 -12 0 -7 -30 -13 21 -12

standard input
4 0 0 2 7 5 8 2 10
standard output
1 2 7 2 10

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

4.4 Number of triangulations

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

Compute the number of triangulations of a convex polygon. Note that this number depends on the number of vertices, but not on the shape of the polygon.

Input

The input contains a single integer n ($3 \leq n \leq 30$), the number of vertices of a convex polygon.

Output

Output the number of triangulations.

Examples

standard input	standard output
6	14
3	1

5 Week 5: Orthogonal range search

5.1 Closest point

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

Given a set S of points *on a line* and a query point, find the closest neighbor of the query point, i.e., a point from S with the minimum distance to the query point.

Input

The first line contains an integer n ($1 \le n \le 50\,000$), the number of points. The second line contains integers $x_1, x_2, \ldots, x_n$, the coordinates of the points.

The third line contains an integer m ($1 \le m \le 50\,000$), the number of query points. Each of the next m lines specifies a query point.

It is guaranteed that each coordinate does not exceed 10^6 by the absolute value.

Output

For each query point, output the nearest point from the given set. In case there are more than one such point, output any single one of them.

Example

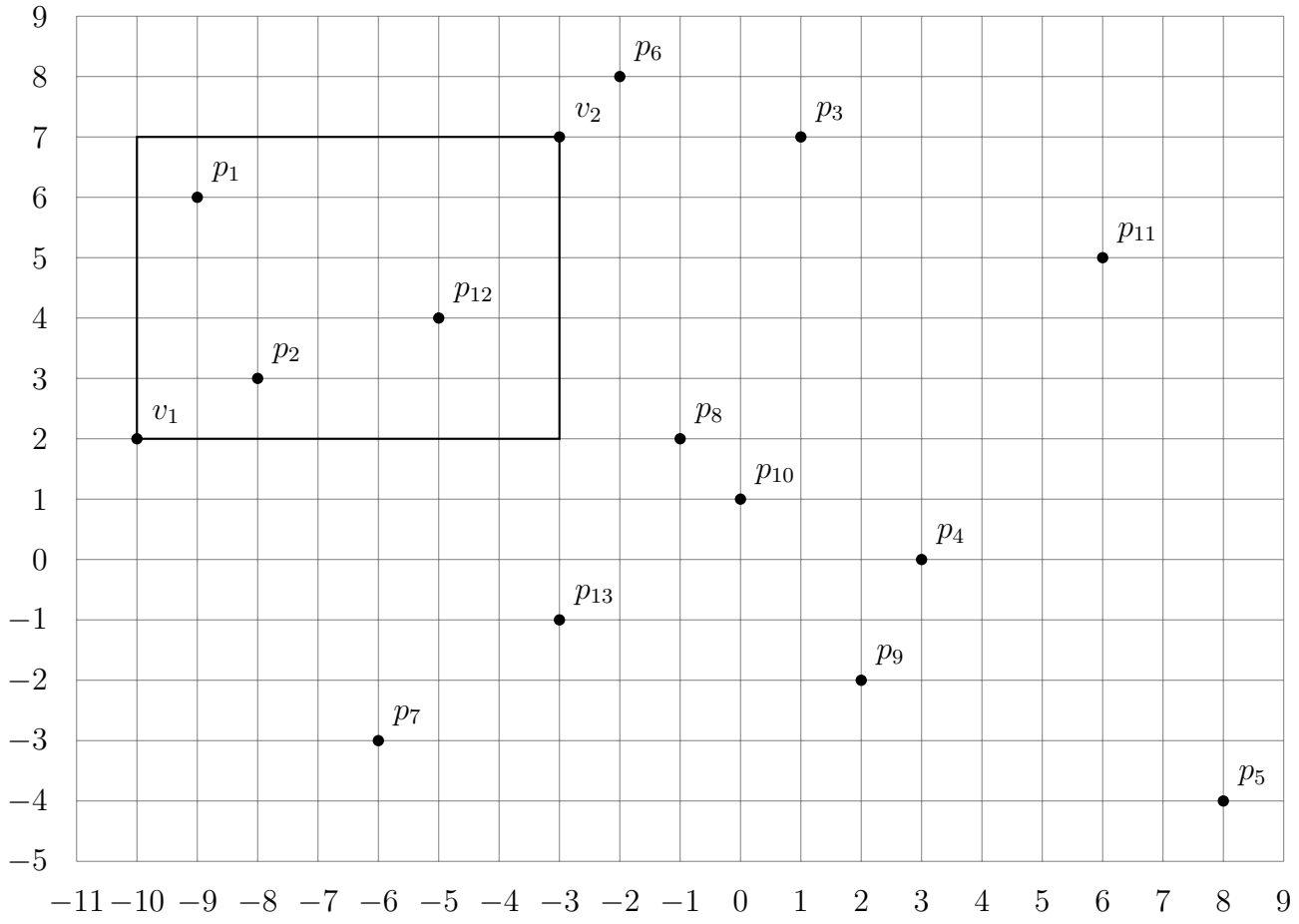
standard input
6 -7 3 8 12 -3 -10 7 -12 13 -5 -3 5 -2 0
standard output
-10 12 -3 -3 3 -3 3

5.2 The number of points in a rectangle

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

Given a set of points S and a rectangle with sides parallel to the axes, compute the number of points from S lying inside the rectangle. Any of the two sides of the rectangle may have zero length (in such a case, the rectangle degenerates to a segment or a point).

For example, a rectangle given by two of its diagonal vertices $v_1(-10, 2)$ and $v_2(-3, 7)$ contains three points from the set S consisting of the following points: $p_1(-9, 6)$, $p_2(-8, 3)$, $p_3(1, 7)$, $p_4(3, 0)$, $p_5(8, -4)$, $p_6(-2, 8)$, $p_7(-6, -3)$, $p_8(-1, 2)$, $p_9(2, -2)$, $p_{10}(0, 1)$, $p_{11}(6, 5)$, $p_{12}(-5, 4)$, $p_{13}(-3, -1)$.



Input

The first line contains an integer n ($1 \le n \le 30\,000$), the number of points.

The second line contains $2n$ integers $p_{1,x}, p_{1,y}, \ldots, p_{n,x}, p_{n,y}$, the coordinates of points. All points are different.

The third line contains an integer q ($1 \le q \le 30\,000$), the number of queries.

The i-th of the following q lines contains four integers $p_{1,x}, p_{1,y}, p_{2,x}, p_{2,y}$, the coordinates of two opposite vertices of a rectangle.

It is guaranteed that each coordinate does not exceed 10^6 by the absolute value.

Output

For each query rectangle, output the number of points in it.

Example

standard input
13 -9 6 -8 3 1 7 3 0 8 -4 -2 8 -6 -3 -1 2 2 -2 0 1 6 5 -5 4 -3 -1 4 -10 2 -3 7 -4 -2 -2 -1 4 2 7 4 -9 -4 8 8
standard output
3 1 0 13

5.3 Closest point

Input file:	`standard input`
Output file:	`standard output`
Time limit:	7 seconds
Memory limit:	256 megabytes

Given a set S of points on the plane and a query point, compute the minimum number of moves needed to get from the query point to a point in S. In a single move, one can go from a point (x, y) to a point $(x + a, y + b)$ where $a, b \in \{-1, 0, 1\}$ (that is, one step horizontally, vertically, or diagonally).

Input

The first line contains an integer n ($1 \le n \le 30\,000$), the number of points.

The second line contains $2n$ integers $p_{1,x}, p_{1,y}, \ldots, p_{n,x}, p_{n,y}$, the coordinates of the points.

The third line contains an integer q ($1 \le q \le 30\,000$), the number of query points. Each of the following q lines specifies a query point.

It is guaranteed that each coordinate does not exceed 10^6 by the absolute value.

Output

For each query point, output its distance to the set of points, i.e., the minimum number of moves needed to get to the set.

Example

standard input
13 -9 6 -8 3 1 7 3 0 8 -4 -2 8 -6 -3 -1 2 2 -2 0 1 6 5 -5 4 -3 -1 5 0 2 3 2 6 5 10 0 -2 10
standard output
1 2 0 4 2

5.4 Queries on segments

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

Implement a program for efficient processing of the following queries involving segments on a line:

- `Add`: add segment to the set;
- `Delete`: delete segment from the set;
- `Count`: compute the number of segments of the set intersecting the given segment.

.

For each query of type "Count", output the number of segments that cross the given one.

Input

The first line contains an integer q ($1 \le q \le 30\,000$), the number of requests.

The i-th of the following q lines contains a query type $s \in \{\texttt{Add}, \texttt{Delete}, \texttt{Count}\}$ and two integers p_{x1}, p_{x2} specifying the endpoints of a segment.

Output

For each "Count" query, output the number of segments intersecting the given segment.

Example

standard input	standard output
16 Add -13 -10 Add -9 -5 Add -8 -6 Add -7 -2 Add 0 3 Add 2 5 Count -14 -1 Count -12 -11 Count -1 6 Count -13 -10 Delete -8 -6 Count -14 -1 Delete 0 3 Count -1 6 Delete -13 -10 Count -13 -10	4 1 2 1 3 1 0

6 Appendix: Compiler Flags

The following compilers and flags are used while grading your solutions.

C (`gcc 5.2.1`). File extensions: `.c`. Flags:
```
gcc -pipe -O2 -std=c11 <filename> -lm
```

C++ (`g++ 5.2.1`). File extensions: `.cc`, `.cpp`. Flags:
```
g++ -pipe -O2 -std=c++14 <filename> -lm
```

C# (`mono 3.2.8`). File extensions: `.cs`. Flags:
```
mcs
```

Haskell (`ghc 7.8.4`). File extensions: `.hs`. Flags:
```
ghc -O2
```

Java (`Open JDK 8`). File extensions: `.java`. Flags:
```
javac -encoding UTF-8
java -Xmx1024m
```

JavaScript (`Node v6.3.0`). File extensions: `.js`. Flags:
```
nodejs
```

Python 2 (`CPython 2.7`). File extensions: `.py2` or `.py` (a file ending in `.py` needs to have a first line which is a comment containing "python2"). No flags:
```
python2
```

Python 3 (`CPython 3.4`). File extensions: `.py3` or `.py` (a file ending in `.py` needs to have a first line which is a comment containing "python3"). No flags:
```
python3
```

Ruby (`Ruby 2.1.5`). File extensions: `.rb`. No flags:
```
ruby
```

Scala (`Scala 2.11.6`). File extensions: `.scala`. No flags:
```
scalac
```