TASK OVERVIEW SHEET / DAY-1

TASK		Hermes	Artemis	Polygon
Time limit per test		1 second CPU	1 second CPU	-
Memory limit		16 MB	16 MB	-
Compiler options	**C**	`-pipe –static –O2 –DCONTEST –lm`		-
	C++	`-pipe –static –include /usr/include/stdlib.h –O2 -DCONTEST –lm`		-
	Pascal	`-So –O1 –XS –dCONTEST`		-
Number of tests		20	20	10
Maximum points per test		5	5	10
Maximum total points		100	100	100

Program header comment when using C		`/*`
		`TASK: hermes`
		`LANG: C`
		`*/`
			`/*`
			`TASK: artemis`
			`LANG: C`
			`*/`
				-

Program header comment when using C++		`/*`
		`TASK: hermes`
		`LANG: C++`
		`*/`
			`/*`
			`TASK: artemis`
			`LANG: C++`
			`*/`
				-

Program header comment when using Pascal		`(*`
		`TASK: hermes`
		`LANG: PASCAL`
		`*)`
			`(*`
			`TASK: artemis`
			`LANG: PASCAL`
			`*)`
				-

Submission is accepted if:		Example input is solved.	Example input is solved.	The file format is correct.

Artemis

PROBLEM

Zeus gave Artemis, the goddess of the wilderness, a rectangular area for growing a forest. With the left side of the area as a segment of the positive y-axis and the bottom side as a segment of the positive x-axis, and (0,0) the left bottom corner of the area, Zeus told Artemis to plant trees only on integer coordinate points in the area. Artemis liked the forest to look natural, and therefore planted trees in such a way that a line connecting two trees was never parallel to x-axis or y-axis.

At times, Zeus wants Artemis to cut trees for him. The trees are to be cut as follows:

1. Zeus wants at least a given number *T* of trees to be cut for him.
2. To get a rectangular football pitch for future football success, Artemis is to cut all trees within a rectangular area, and no trees outside of it.
3. The sides of this rectangular area are to be parallel to x-axis and y-axis.
4. Two opposite corners of the area must be located on trees and therefore those corner trees are also cut.

As Artemis likes the trees, she wants to fulfill these conditions whilst cutting as few trees as possible. You are to write a program that, given information on the forest and the minimum number *T* of trees to be cut, selects an area for cutting trees for Artemis.

INPUT

The input file name is `artemis.in`. The first line contains one integer *N*: the number of trees in the forest. The second line contains one integer *T*: the minimum number of trees to be cut. The following *N* lines describe the positions of the *N* trees. Each of these lines contains two integers X and Y: the x-coordinate followed by the y-coordinate of a tree.

OUTPUT

The output file name is `artemis.out`. The file is to contain one line with two integers *I* and *J* separated by one space: Artemis should use the *I*th tree (with position given on line *I*+2 of the input file) and *J*th tree (with position given on line *J*+2 of the input file) as the corners of the area for cutting trees. The order of these two number is irrelevant. There may be several ways to choose these trees and you need to find and output one of them. For all test cases at least one solution exists.

EXAMPLE INPUT AND OUTPUT

`artemis.in`

```
3
2
1 1
2 3
5 6
```

`artemis.out`

```
1 2
```

CONSTRAINTS

In all inputs, $1<N\leq 20000$, $0\leq X,Y\leq 64000$ and $1<T\leq N$.
Additionally, in 50 % of the inputs, $1<N<5000$.

Hermes

PROBLEM

In a modern city for Greek gods, the streets are geometrically arranged as a grid with integer coordinates with streets parallel to the x and y axes. For each integer value *Z*, there is a horizontal street at y=*Z* and a vertical street at x=*Z*. This way, integer coordinate pairs represent the street junctions. During the hot days, the gods rest in cafeterias at street junctions. Messenger Hermes is to send photon messages to gods resting in the cafeterias by only moving along the city streets. Each message is for a single god, and it does not matter if the other gods see the message.

The messages are to be sent in a given order, and Hermes is provided the coordinates of the cafeterias in that order. Hermes starts from (0,0). To send a message to a cafeteria at (X_i,Y_i), Hermes only needs to visit some point on the same horizontal street (with y-coordinate Y_i) or on the same vertical street (with x-coordinate X_i). Having sent all of the messages, Hermes stops.

You are to write a program that, given a sequence of cafeterias, finds the minimum total distance Hermes needs to travel to send the messages.

INPUT

The input file name is `hermes.in`. The first line contains one integer *N*: the number of messages to be sent. The following *N* lines contain the coordinates of the *N* street junctions where the messages are to be sent. These *N* lines are in the order in which the messages are to be sent. Each of these *N* lines contains two integers: first the x-coordinate and then the y-coordinate of the street junction.

OUTPUT

The output file name is `hermes.out`. The file is to contain a single line containing one integer: the minimum total distance Hermes needs to travel to send the messages.

EXAMPLE INPUT AND OUTPUT

`hermes.in`

```
5
8 3
7 -7
8 1
-2 1
6 -5
```

`hermes.out`

```
11
```

CONSTRAINTS

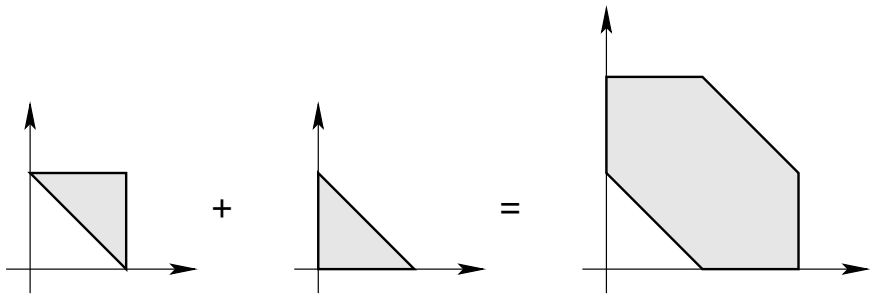
In all inputs, $1 \le N \le 20000$, $-1000 \le X_i, Y_i \le 1000$. Additionally, in 50% of the inputs, $1 \le N \le 80$.

Polygon

PROBLEM

A polygon consists of all points on or enclosed by its border. A convex polygon has the property that for any two points *X* and *Y* of the polygon, the line segment connecting *X* and *Y* is inside the polygon. All polygons in this task are convex polygons with at least two vertices, and all vertices in a polygon are different and have integer coordinates. No three vertices of the polygon are collinear. The word "polygon" below always refers to such polygons.

Given two polygons *A* and *B*, the Minkowski sum of *A* and *B* consists of all the points of the form (x_1+x_2, y_1+y_2) where (x_1, y_1) is a point in *A* and (x_2, y_2) is a point in *B*. It turns out that the Minkowski sum of polygons is also a polygon. The figure below shows an example: two triangles and their Minkowski sum.



We study a reverse operation to the Minkowski sum. For a given polygon *P*, we are looking for two polygons *A* and *B* such that:

- *P* is the Minkowski sum of *A* and *B*,
- *A* has from 2 to 4 different vertices, i.e. it is a segment (2 vertices), a triangle (3 vertices) or a quadrilateral (4 vertices),
- *A* should have as many vertices, as possible, i.e.:
  - *A* should be a quadrilateral, if possible,
  - if *A* cannot be a quadrilateral, it should be a triangle, if possible,
  - otherwise it should be a segment.

Clearly, neither *A* nor *B* can be equal to *P* because then the other summand would have to be a point, which is not a valid polygon.

You are given a set of input files, each containing a description of a polygon *P*. For each input file you should find the polygons *A* and *B*, as required above, and create an output file containing descriptions of *A* and *B*. For the given input files such polygons *A* and *B* can always be found. If there are many correct results, you should find and output one of them. You should not submit any programs, just the output files.

INPUT

You are given 10 problem instances in the text files named `polygon1.in` to `polygon10.in`, where the number after `polygon` is the input number. Each input file is

organized as follows. The first line contains one integer N: the number of vertices of the polygon P. The following N lines describe the vertices in a counter-clockwise order, one vertex per line. Line I+1 (for I = 1, 2, ..., N) contains two integers X_I and Y_I, separated by a space: coordinates of the Ith vertex of the polygon. All input coordinates are non-negative integers.

OUTPUT

You are to submit 10 output files corresponding to the given input files which describe the required polygons A and B. The first line is to contain the text:
`#FILE polygon I`
where integer `I` ($1 \le$ `I` ≤ 10)is the number of the respective input file.

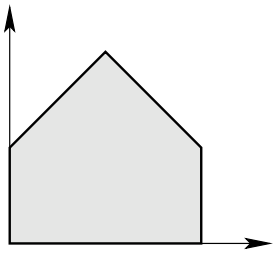
The output format is similar to the input format. The second line is to contain one integer N_A: the number of vertices in A ($2 \le N_A \le 4$). The following N_A lines describe the vertices of A in the counter-clockwise order, one vertex per line. Line I+2 (for I = 1, 2, ..., N_A) contains two integers X and Y, separated by a space: coordinates of the Ith vertex of the polygon A.

Line N_A+3 should contain one integer N_B: the number of vertices in B, ($2 \le N_B$). The following N_B lines describe the vertices of B in the counter-clockwise order, one vertex per line. Line N_A+J+3 (for J = 1, 2, ..., N_B) contains two integers X and Y, separated by a space: coordinates of the Jth vertex of the polygon B.

EXAMPLE INPUT AND OUTPUTS

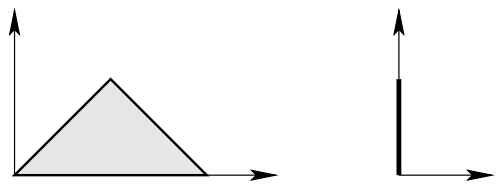
`polygon0.in`

```
5
0 1
0 0
2 0
2 1
1 2
```



For the above input, either of the below output files (see also the figures) is correct, since in both cases A is a triangle and it cannot be a quadrilateral.

```
#FILE polygon 0
3
0 0
2 0
1 1
2
0 1
0 0
```



```
#FILE polygon 0
3
0 0
1 0
1 1
3
0 1
0 0
1 0
```