(Adv.) Competitive Programming

Submit until 05.07.2019 13:30, via the judge interface

Problem: sheep2 (1 second timelimit)

Note: This problem is split into two parts: easy and hard. They differ only by the limits on the problem input. Solving both parts counts as solving the full problem, solving only easy counts as half.

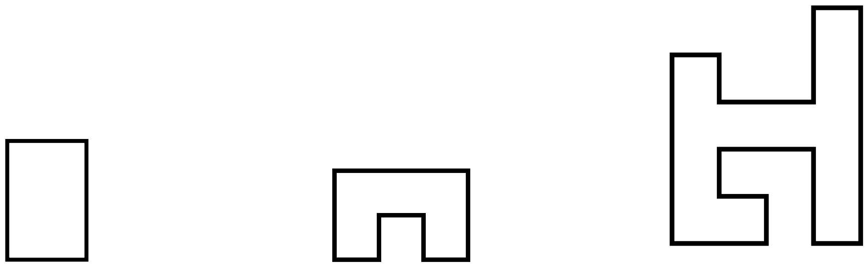
As part of your new job at HPI, you are responsible for the software of the famous sheep lawn mower. Optimizing its routing algorithm was somewhat of a sport for your predecessor. And while his algorithm produces impressive results, it has one fatal flaw: sometimes the battery runs out while mowing.

Fixing this bug is your first assignment. Unfortunately, it seems that code quality wasn't high on the priority list. Endless hours of debugging later, some of which you literally spend watching the sheep move slowly around the lawn, you think that you've understood what the algorithm does. It moves the sheep around seemingly at random, however, only using 90° angle turns. When the sheep reaches a part is has already mowed, it then stops to mow the part enclosed by its path.

The calculation that estimates the power requirements for the latter part seems to be incomplete, and thus causes the sheep to run out of battery. To fix it, you will need to calculate the area of the enclosed part.

Input The input contains the enclosed part of the lawn as a polygon containing only 90° angle turns. It begins with a line containing n (for easy: $4 \leq n \leq 5000$, for hard: $4 \leq n \leq 200\,000$), the number of points. The next n lines contain the points in counter-clockwise order (assuming the x-axis points right and the y-axis up). Each point is given as x_i and y_i ($-10^9 \leq x_i, y_i \leq 10^9$). For each two adjacent points, either their x or y coordinates are equal, but never both. No three adjacent points form a line.

Output Output the area of the polygon.



Visualizations of the samples

Sample input

```
4
0 0
2 0
2 3
0 3
```

Sample output

```
6
```

Sample input

```
8
3 2
0 2
0 0
1 0
1 1
2 1
2 0
3 0
```

Sample output

```
5
```

```
14
0 0
2 0
2 1
1 1
1 2
3 2
3 0
4 0
4 5
3 5
3 3
1 3
1 4
0 4
```

```
12
```