H: Ghost Hunting

Source file name: `hunting.c`, `hunting.cpp`, `hunting.java`, *or* `hunting.py`
Author: Germán Sotelo

A ghost Pokemon gang has been playing pranks on citizens and damaging public property in Lavender City. Any attempt to apprehend the gang has been futile, since it is impossible to see a ghost with the naked eye.

The major of Lavender City requested help from Silph Corp. After several months of research, the corporation developed a system that allows people to see the ghost Pokemon. The system is based on beacons: when a ghost Pokemon is surrounded by them, it is visible to anyone. However, the beacons use a lot of energy and thus require to be installed in light poles already available in the city. A Pokemon is *surrounded by beacons* if, in order to leave town, it needs to cross through a virtual fence: the idea is that any pair of beacons induce a virtual fence defined by the segment of a straight line starting at one of them and ending at the other one.

The past week, Silph Corp sent a truck full of beacons to create a visibility area in Lavender City. However, the ghost gang attacked the truck and damaged most of the beacons, except for three that are still fully operational. Since the production of beacons takes time, and the situation is getting dangerous by the day, the major wants to find the largest possible area that can be set up to see the ghosts once the three beacons are installed.

Given the location of the light poles, your help has been requested to find such an area.

Input

The input consists of several test cases. Each test case begins with a line containing an integer N ($3 \leq N \leq 2 \times 10^3$) representing the number of light poles in Lavender City. Each of the following N lines contains two blank-separated integers x and y ($-10^6 < x, y < 10^6$) indicating the location (x, y) of a light pole.

The input must be read from standard input.

Output

For each test case, output a single line with the largest possible area, truncated to one decimal, where the ghosts can be seen.

The output must be written to standard output.

Sample Input	**Sample Output**
4 -1 -1 -1 1 1 -1 1 1 3 -1 -1 -1 3 3 3 4 1 1 2 1 4 1 4 6	2.0 8.0 7.5