# Problem 3 – Critical Breakpoint

Imagine a plane (a dimension) and parallel lines in it. Normally they would never meet, but let us imagine that we are working in magical planes and all of them have a point where they bend infinitely in order to connect to the other planes. That point is called the **critical breakpoint**.

You are given a set of **4 integers** separated by a single space, on every line of input, which represents plane coordinates for each line – {x1, y1}, {x2, y2}, the two points that form the line. When you get the command “Break it.”, you **END** the input. Your task is to find if the given lines form a plane which holds a critical breakpoint.

The critical breakpoint is formed **only** if the **critical ratio** of the current line is **equal to the others’ critical ratio** or is equal to **ZERO**. The critical ratio is, for each line, **the absolute value** of “(X2 + Y2) – (X1 + Y1)”.

If you find a **non-zero** critical ratio value it becomes the **actual** critical ratio, and if there is even one line, which’s critical ratio does not equal that value, **and** is not equal to zero, the current lines fail to create a critical breakpoint. Then you print “Critical breakpoint does not exist.”.

If a critical breakpoint is formed you need to calculate it. A critical breakpoint is equal to – the **remainder** of the **division** of, the **critical ratio** (which in this case is one for all the lines) **powered** by the count of lines, **and the count of lines**. If we have 4 lines and criticalratios – “10, 0, 0, 10” the breakpoint will be the **remainder** of: “(104) / 4”, which is **0**.

### Input

* The input will come in the form of lines on which you will get a set of 4 integers separated by a single space.
* The input continues until you receive the command “Break it.”.

### Output

* If a critical breakpoint has been found you need to print each of the lines in the following format:
* “Line: [x1, y1, x2, y2]”
* And after them, print the critical breakpoint in this format:
* “Critical Breakpoint: {breakpoint}”
* In case there is no breakpoint you must print only the following:
* “Critical breakpoint does not exist.”

### Constraints

* All of the input numbers will be valid integers in the range [-231 + 1, 231 - 1].
* Allowed time/memory: 250ms/16MB.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comment** |
| 1 2 3 4  2 3 4 1  3 4 1 2  4 1 2 3  Break it. | Line: [1, 2, 3, 4]  Line: [2, 3, 4, 1]  Line: [3, 4, 1, 2]  Line: [4, 1, 2, 3]  Critical Breakpoint: 0 | First line: (4 + 3) – (2 + 1) = 4  Second Line: (4 + 1) – (2 + 3) = 0  Third Line: (1 + 2) – (4 + 3) = -4 = 4(Absolute)  Fourth Line: (2 + 3) – (4 + 1) = 0  The lines either have critical ratio – 4 or 0, thus, they **form** a critical breakpoint, which is:  44/ 4 =  256 / 4 = 64, **remainder 0** |

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2 3 12 0  3 12 16 0  Break it. | Critical breakpoint does not exist. |