Practice Arena

Practice problems aimed to improve your coding skills.

- PRACTICE-02_SCAN-PRINT
- PRACTICE-03_TYPES
- LAB-PRAC-02_SCAN-PRINT
- LAB-PRAC-01
- PRACTICE-04_COND
- BONUS-PRAC-02
- LAB-PRAC-03_TYPES
- PRACTICE-05_COND-LOOPS
- LAB-PRAC-04_COND
- LAB-PRAC-05_CONDLOOPS
- PRACTICE-07_LOOPS-ARR
- LAB-PRAC-06_LOOPS
- LAB-PRAC-07_LOOPS-ARR
- LABEXAM-PRAC-01_MIDSEM
- PRACTICE-09_PTR-MAT
- LAB-PRAC-08_ARR-STR
- PRACTICE-10_MAT-FUN
- LAB-PRAC-09_PTR-MAT
- LAB-PRAC-10_MAT-FUN
- PRACTICE-11_FUN-PTR
- LAB-PRAC-11_FUN-PTR
- LAB-PRAC-12_FUN-STRUC
    - Point Pairing Party
    - Verify the family tree of Mr C
    - Simple Sodoku
    - The Family Tree of Mr C Part Three
    - The Post offices of KRville
    - Matrix Mandala
    - Mango Mania
    - Recover the Rectangle
    - Crazy for Candy
    - A Brutal Cipher Called Brutus
    - Triangle Tangle
    - Basic Balanced Bracketing
- LABEXAM-PRAC-02_ENDSEM
- LAB-PRAC-13_STRUC-NUM
- LAB-PRAC-14_SORT-MISC

Point Pairing Party

LAB-PRAC-12_FUN-STRUC

Point Pairing Party [20 marks]

Problem Statement
The first line of the input will give you n, a strictly positive integer. The next n lines will give you n pairs of points on the 2D plane. Each point will have two integer coordinates, separated by a space. The two points in the pair will also be separated by a space.

Naturally, each pair of these points corresponds to a unique line which passes through both the points. In your output you have to print how many pairs of these points correspond to lines that are parallel, that intersect at a single point, and that are identical (i.e. infinite points of intersection). Give your output in the format described below.

HINT: You would find it a bit cumbersome to manage the coordinates for all the points in separate arrays. Instead, you should use structures to make your code easier to read as well as easier to debug. Define a structure for a 2D point which stores two integers, corresponding to the x and y coordinates of the point. Define another structure for a pair of 2D points. Now you can define an array of these pairs and work with them. An example is given below

```
struct point{
   int x, y
};

struct pair{
   struct point p1, p2;
};
```

Caution

1. The question may require you to compare floating point numbers like slopes etc. Since comparing floating point numbers for equality is dangerous, we will consider two floating point numbers to be the same if their difference in absolute terms is less than 0.0001. Use the fabs() function from math.h to get the absolute value of floating point numbers.
2. Do not make spelling, capitalization or space errors. Look at the output format closely below.
3. Be careful about divide-by-zero errors. Some of the lines may be horizontal or vertical.
4. Count each pair only once. Do not make double counting errors.
5. We assure you that there will be no pair of points were both points are the same since such pairs do not correspond to a unique line.
6. Be careful about extra/missing lines and extra/missing spaces in your output.

EXAMPLE:
INPUT

```
2
1 1 0 0
1 0 1 4
```

OUTPUT:

```
PARALLEL: 0
INTERSECT: 1
IDENTICAL: 0
```

Explanation: The first line passes through the points (1, 1) and (0, 0); the second line passes through the points (1, 0) and (1, 4). Both the lines intersect at (1, 1). Thus, there are no pairs of parallel lines, 1 pair of intersecting lines and no pairs of identical lines.

Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. There are three lines in your output. Printing each line correctly, in the correct order, carries 33% weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

Start Solving! (/editor/practice/6230)