Two Circles

Problem Statement

You are given two circles, *A* and *B*, on a Cartesian plane, each defined by three descriptors:

1. *X*: the x-coordinate of the circle's center
2. *Y*: the y-coordinate of the circle's center
3. *R*: the radius of the circle

Circles *A* and *B* will both be centered *either* on the *X-axis* (i.e.: $Y_A = 0$ and $Y_B = 0$), *or* on the *Y-axis* (i.e.: $X_A = 0$ and $X_B = 0$).

A pair of circles (*A* and *B*) will have one of the following relationships types:

a. `Touching`: they touch each other at a single point.
b. `Concentric`: they have the same center point.
c. `Intersecting`: they intersect each other (touching at two points).
d. `Disjoint-Outside`: disjoint with one existing outside of the other.
e. `Disjoint-Inside`: disjoint with one contained inside the other (but not concentric).

Complete the *circles* function which takes an array of strings, *info*, as its parameter. Each string element in *info* contains six space-separated integers denoting a test case (as shown in the *Input Format*). The function should return an array of strings where the i^{th} element in the return array is the relationship for the circles defined in the i^{th} element of *info*.

Input Format

The first line contains an integer, *N* (the number of test cases).
The *N* subsequent lines of test cases each contain six space-separated integers describing the *X*, *Y*, and *R* values for circles *A* and *B*, respectively. For example:

$$\begin{array}{l} N \\ X_{A_0}\ Y_{A_0}\ R_{A_0}\ X_{B_0}\ Y_{B_0}\ R_{B_0} \\ \ldots \\ X_{A_{N-1}}\ Y_{A_{N-1}}\ R_{A_{N-1}}\ X_{B_{N-1}}\ Y_{B_{N-1}}\ R_{B_{N-1}} \end{array}$$

Note: Reading input from stdin and calling *circles* is handled for you by the locked code in the editor. Your task is to process the array of input strings in *circles*.

Output Format

Your *circles* function should return an array of *N* strings where the i^{th} element is the relationship for the circles in *info[i]*. Recall that the relationships defined in the *Problem Statement* are `Touching`, `Disjoint-Inside`, `Disjoint-Outside`, `Concentric`, and `Intersecting`.

Note: Outputting the array returned by *circles* is handled for you by the locked code in the editor.

Sample Input 0

```
4
12 0 21 14 0 23
0 45 8 0 94 9
35 0 13 10 0 38
0 26 8 0 9 25
```

Sample Output 0

```
Touching
Disjoint-Outside
Touching
Touching
```

Sample Input 1

```
5
0 5 9 0 9 7
0 15 11 0 20 16
26 0 10 39 0 23
37 0 5 30 0 11
41 0 0 28 0 13
```

Sample Output 1

```
Intersecting
Touching
Touching
Intersecting
Touching
```

We recommend you take a quick tour of our editor before you proceed.
The timer will pause up to 90 seconds for the tour. Start tour

5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23

Original code | Java 7

```
import ↔;

public class Solution {


/*
 * Complete the function below.
 */

    static String[] circles(String[] info) {


    }


    public static void main(String[] args) throws IOException{↔}
}
```

Line: 12 Col: 1

Run Code | Submit code & Continue

(You can submit any number of times)

Test against custom input

Download sample test cases *The input/output files have Unix line endings. Do not use Notepad to edit them on windows.*