

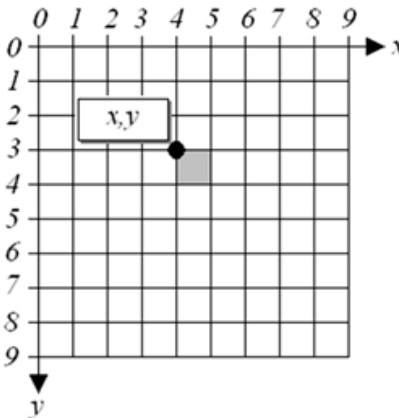
2D Rectangle Collision

Filename: collision

The Problem

Collision detection is necessary in many 2-Dimensional games. For example, if the player is trying to shoot objects, the player would need to know whether the object has been hit or not. 2D rectangle collision is one simple technique for accomplishing just that. If you think of the bullet a player is firing as a rectangle, and the enemy the player is firing at as another rectangle, all you need to do is determine if the two rectangles intersect. Your job is to create a method to test the intersection of two objects. You will be given each object's position, and its width and height.

Remember that you will be using the pixel coordinate system. This means that the top left corner of the screen is (0, 0), where X increases to the right, and Y increase down. The (x, y) coordinates of the point below are (4, 3). When we specify a rectangle, four pieces of information specify where it will be in the pixel coordinate system. The associated (x, y) coordinate represents the top-left hand corner of the rectangle in question. The width of the rectangle represents how long the horizontal sides of the rectangle are and the height of the rectangle represents how long the vertical sides of the rectangle are. (For example, a rectangle with x-y coordinates (4,3), width 2 and height 7 would have bottom right-hand coordinates of (6, 10).)



In order for two rectangles to be considered intersecting, they must share at least one square unit.

The Input

The first line of the input file will contain a single positive integer, n , representing the number of input test cases. The following n lines will contain 8 positive integers, the first 4 integers on each line will pertain to object 1 and the second 4 will pertain to object 2. The 4 integers will be the X position, the Y position, the width W , and the height H , where X , Y , W , and H are between 0 and 999. You must determine whether object 1 collides with object 2.

The Output

For each test case, output a single line with one of the following formats, based on whether or not the objects collide:

Case #X: Object 1 COLLIDES with Object 2.

Case #X: Object 1 DOES NOT COLLIDE with Object 2.

where X is the 1-based case number.

Sample Input

```
2
100 100 200 100 175 25 76 90
12 2 4 3 5 5 7 3
```

Sample Output

Case #1: Object 1 COLLIDES with Object 2.

Case #2: Object 1 DOES NOT COLLIDE with Object 2.