Overlapping 2D shapes

Submission deadline:	**2011-10-30 23:59:59**	1078671.652 sec
Evaluation:	**0.0000**	
Max. assessment:	**5.0000** (Without bonus points)	
Submissions:	0 / 10 Free retries + 20 Penalized retries (-2 % penalty each retry)	
Advices:	0 / 2 Advices for free + 2 Advices with a penalty (-10 % penalty each advice)	

Your task is to develop a program which analyzes position of 2D shapes (circles and rectangles) in a 2D plane.

The input of your program are two geometrical shapes. A shape may be either a rectangle or a circle. A rectangle is described by a pair of opposite vertices - a 2D coordinate is given for both of them. Note, the vertices are guaranteed opposite, however, they may be swapped and they may be left-lower + right-upper as well as left-upper + right-lower. The sides of the rectangle are parallel to the x,y axes. A circle is described using a center point coordinate and a radius. All numbers are integers. The exact input format is shown in samples below.

The output of the program is a decision whether the two shapes intersect, or not. The exact output format is shown in sample runs below. If the two shapes share at least one common point, the intersection is considered nonempty.

If the input is invalid, the program detects the problem and prints an error message. The format of the error message is again shown below. The following is considered invalid input:

- non-numeric value,
- invalid shape identification (R/C),
- negative or zero length radius,
- zero length of rectangle side.

If the program detects an error, it immediately stops asking for further input data, it prints out the error message, and terminates. Thus, the program must validate input data as it reads them. Do not postpone the checks until entire input is read. The error message shall be printed on the standard output (do not send it to the standard error output).

Please strictly adhere the format of the output. The output must exactly match the output of the reference program. The comparison is done by a machine, the machine requires an exact match. If your program provides output different from the reference, the program is considered malfunctioning. Be very careful, the machine is sensitive event to whitespace characters (spaces, newlines, tabulators). Please note that all output lines are followed by a newline character (\n). This applies even to the last line of the output, moreover, this applies even to the error message. Download the enclosed archive. The archive contains a set of testing inputs and the expected outputs. Read FAQ to learn how to use input/output redirection and how to simplify testing of your programs.

Your program will be tested in a restricted environment. The testing environment limits running time and available memory. The exact time and memory limits are shown in the reference solution testing log. However, neither time nor memory limit could cause a problem in this simple program.

Sample program output:

```
Shape 1 (R=rectangle, C=circle):
R
First vertex:
1 1
Second vertex:
5 5
Shape 2 (R=rectangle, C=circle):
R
First vertex:
7 3
Second vertex:
3 7
```

```
Nonempty intersection.
```

```
Shape 1 (R=rectangle, C=circle):
R
First vertex:
0 0
Second vertex:
10 10
Shape 2 (R=rectangle, C=circle):
C
Center point:
5 5
Radius:
20
Nonempty intersection.
```

```
Shape 1 (R=rectangle, C=circle):
C
Center point:
0 0
Radius:
5
Shape 2 (R=rectangle, C=circle):
C
Center point:
0 10
Radius:
5
Nonempty intersection.
```

```
Shape 1 (R=rectangle, C=circle):
C
Center point:
0 0
Radius:
5
Shape 2 (R=rectangle, C=circle):
R
First vertex:
0 20
Second vertex:
10 30
Empty intersection.
```

```
Shape 1 (R=rectangle, C=circle):
A
Invalid input.
```

```
Shape 1 (R=rectangle, C=circle):
R
First vertex:
0 10
Second vertex:
0 20
Invalid input.
```

```
Shape 1 (R=rectangle, C=circle):
C
```

```
Center point:
40 50
Radius:
-4
Invalid input.
```

```
Shape 1 (R=rectangle, C=circle):
R
First vertex:
1 abcd
Invalid input.
```

Sample data: **Download**

Submit: [redacted] **Submit**

☐ **Reference**