COS 126 Programming Exam 2 Fall 2018

This exam is like a mini programming assignment. You will have 50 minutes to create and submit three programs. Debug your code as needed. You may use your book, your notes, your code from programming assignments, the code on the COS 126 course website, the booksite, and you may read Piazza. No form of communication is permitted (e.g., talking, email, texting, etc.).

Downloads. Before you begin (now), download the Project file on the Meetings page.

Submissions. Submit your programs using the submit links for Programming Exam 2 on the Meetings page. Submit Part 1 before attempting Part 2; submit Part 2 before attempting Part 3.

Grading. Your program will be graded on correctness, clarity (including comments), design, and efficiency. You will lose a substantial number of points if you do not follow submission instructions precisely, or if your program does not compile or if it crashes on typical inputs.

Discussing this exam. Due to travel for extracurriculars and sports, some of your peers will take this exam next week. Do not discuss exam contents with anyone who has not taken the exam.

This paper. In addition to submitting your code electronically, you must return this paper. Fill in the information below, then transcribe and sign the Honor Code pledge.

Name	
NetID	
Precept	
Exam Room	

"I pledge my honor that I have not violated the Honor Code during this examination."

_____

_____

[copy the pledge here]

_____

[signature]

Part 1 (10 points): Test cellphone coverage

Your task is to implement a (very) simple model for computing cellphone coverage. The first step is to write a class `CellTower` that implements the following API:

`public class CellTower`	
`CellTower(double x, double y, double r)`	create a new tower at (x,y) with range r
`boolean inRange(double x0, double y0)`	true if (x0,y0) is in range of this tower, false otherwise

We assume that the range of the tower is a circle of radius r centered at the tower position. Therefore, to implement `inRange()`, simply compute the distance between the given point and the tower position and test that it is smaller than the range of the tower.

Specifically, use this test: $\sqrt{(x_0 - x)^2 + (y_0 - y)^2} < r$.

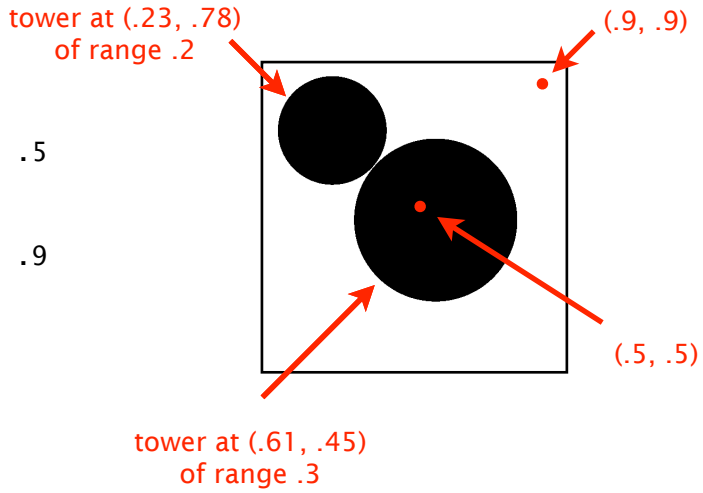
You *must* include a test client `main()` that proceeds as follows:

- Create a tower at (.61, .45) of range .3 .
- Create another tower at (.23, .78) of range .2 .
- Take `double` values `x` and `y` from the command line and print `Covered` if the point (*x*,*y*) is in the range of one or both of those towers and `Not covered` if the point (*x*,*y*) is not in the range of either tower.

The diagram at right illustrates the coverage of these towers, and the required behavior of your program is shown at left.

```
> java-introcs CellTower .5 .5
Covered

> java-introcs CellTower .9 .9
Not covered
```



USE THE SUBMIT LINK ON THE MEETINGS PAGE TO SUBMIT `CellTower.java`.

DO NOT PROCEED TO PART 2 UNTIL YOU HAVE DONE SO.

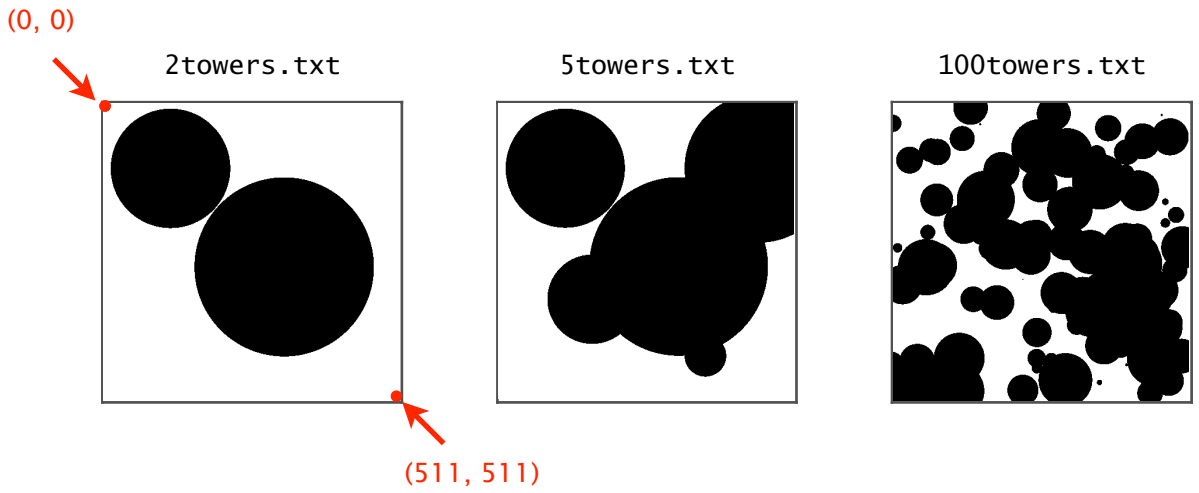
Part 2 (15 points): Visualize cellphone coverage

Implement a `CellTower` and `Queue` client `ShowCoverage` that visualizes the coverage for a set of towers. Assume that the tower data is on standard input, three double values (all between 0 and 1) per line (`x`, `y`, and `r` in that order).

Your program must color black every pixel that corresponds to a point that is in the range of some tower, and color white every pixel that corresponds to a point that is not in the range of any tower. To do this, you need to convert from pixel coordinates to data coordinates: a pixel (i, j) represents the point $(i/512, 1 - j/512)$ in the unit square. Recall that (0, 0) is at the bottom left in the unit square and the top left in a `Picture`. Also, take note that pixels in a `Picture` are all initially black.

Specifically, your program must proceed as follows:

- Create a queue of towers.
- Read the data from standard input and fill the queue with the towers specified by the data.
- Then, build a 512-by-512 `Picture` with pixel `(i, j)` set to `Color.WHITE` if the point `(i/512.0, 1.0 - j/512.0)` is not within the range of any tower, `Color.BLACK` otherwise.
- Show the picture, producing these images for our test files.



USE THE SUBMIT LINK ON THE MEETINGS PAGE TO SUBMIT `ShowCoverage.java`.

DO NOT PROCEED TO PART 3 UNTIL YOU HAVE DONE SO.

Part 3 (5 points): Estimate cellphone coverage

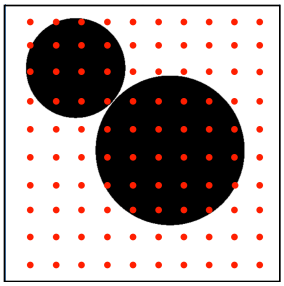
Be sure that you have successfully completed both Part 1 and Part 2 and have submitted your solutions `CellTower.java` *and* `ShowCoverage.java` *before attempting this part of the exam* (which may take longer and counts for many fewer points).

Write a `CellTower` and `Queue` client `CoverageEstimate` that takes an `int` value `T` from the command line and estimates to within one decimal place the percentage coverage of the unit square. As before, put all the towers on a `Queue`. Then proceed by sampling each point in a `T`-by-`T` uniform grid of points, counting the number of points that are within range of some tower in the set of towers on standard input, and dividing by the total number of points.

Center your grid. If the space between each point and its neighbors is $p = 1/T$, then the point at the lower left corner should be at $(p/2, p/2)$.

For example, for `2towers.txt`, the circles are of radius .2 and .3 and do not overlap, so, since the area of a circle of radius r is πr^2, the percentage coverage is

$$\pi(.2)^2 + \pi(.3)^2 \approx 0.4084070449666731$$



For a 10-by-10 uniform grid, there are 39 points inside one or both circles, as shown at right, so your result should be 39/100, or 39.0. For larger values of T, the estimate becomes more accurate:

```
> java-introcs CoverageEstimate 100 < 2towers.txt
40.9 per cent coverage

> java-introcs CoverageEstimate 1200 < 2towers.txt
40.8 per cent coverage
```

For `5towers.txt` and `100towers.txt`, the overlap makes the exact coverage difficult to compute, so sampling is a practical alternative:

```
> java-introcs CoverageEstimate 1200 < 5towers.txt
58.2 per cent coverage

> java-introcs CoverageEstimate 1200 < 100towers.txt
62.8 per cent coverage
```

USE THE SUBMIT LINK ON THE MEETINGS PAGE TO SUBMIT `CoverageEstimate.java`.