Problem 194: Augmenting Reality

Difficulty: Medium

Author: Kelly Reust, Denver, Colorado, United States

Originally Published: Code Quest Community College Outreach 2022

Problem Background

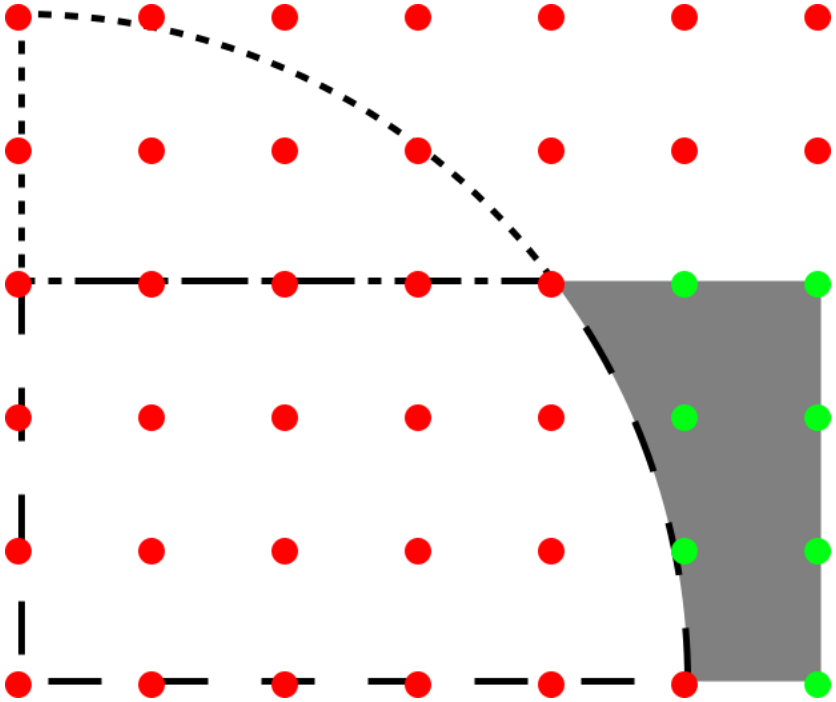
Lockheed Martin is working on a new augmented reality system to help soldiers identify parts to be installed on military aircraft undergoing maintenance. Maintainers will wear a headset that includes a camera and two small screens in the form of eyeglasses. When a maintainer picks up a part, the system will use the images captured by the camera to attempt to identify the part. If the part is successfully identified, a small overlay will appear on the screens providing information about the part in question. From the maintainer's perspective, the overlay will appear to be "hovering" over the part they're holding, allowing them to quickly identify the correct parts to install without having to look up serial numbers on a conventional computer system.

However, your team's goal is to be helpful to the maintainers, and not hinder their work. The overlays you create need to be large enough to include any information they might need, but shouldn't be so large that they cover the user's entire field of vision. Your team has decided to define an area of the screen that should always remain clear to avoid blinding the user; any overlay information must be displayed outside of this area.

Problem Description

The "no-overlay" zone for your team's AR system will be defined as a circle, centered on the lower left corner of the screen. The overlays themselves will be rectangles, positioned so their lower left corner is in the lower left corner of the screen. Any portion of the overlay that falls within the circular "no-overlay" zone should not be rendered. Your team needs to write a program that determines which pixels should be rendered to avoid infringing on the "no-overlay" zone.

For example, see the example screen layout displayed on the next page. This shows a situation where the "no-overlay" zone has a radius of 5 pixels, and the overlay is 6 pixels wide by 3 pixels high.



The curved dotted line represents the border of the "no-overlay" zone. Since it's defined by a circle centered on the screen's lower left corner, we only see the upper right quadrant of the circle. The overlay is represented by the shaded area; even though it's defined by a rectangle anchored to the screen's lower left corner, we only see the portions that extend beyond the "no-overlay" zone. The colored dots represent the pixels, positioned at the integer X,Y coordinates on a grid where the origin is the lower-left corner. The green dots are the only pixels within the rectangular overlay and strictly outside of the circular "no-overlay" zone. The pixels at 4,3 and 5,0 appear on the edge of the circle, and so are not rendered.

Given the radius of the "no-overlay" zone and the dimensions of the overlay's rectangle, your team needs to develop an algorithm to determine which pixels on the screen need to be rendered. Pixels are represented by non-negative integer X,Y coordinates, where the origin (0,0) is the lower left corner of the screen. Higher X values appear further to the right on the screen, and higher Y values appear further up on the screen.

Sample Input

The first line of your program's input, received from the standard input channel, will contain a positive integer representing the number of test cases. Each test case will include a single line with the following values, separated by spaces:

- R, a positive number representing the radius of the circular "no-overlay" zone, centered on the lower-left corner of the screen
- W, a positive number representing the width of the overlay rectangle, as measured from the lower-left corner of the screen

- H, a positive number representing the height of the overlay rectangle, as measured from the lower-left corner of the screen

```
2
5 6 3
4 5 3
```

Sample Output

For each test case, your program must print a list of non-negative integer X,Y coordinates that fall within the defined rectangle but do not fall within the defined circle. Each coordinate should be printed on a separate line, and should be sorted in increasing order; first by the X coordinate, then by the Y coordinate. Numbers should be separated by a comma.

```
5,1
5,2
5,3
6,0
6,1
6,2
6,3
3,3
4,1
4,2
4,3
5,0
5,1
5,2
5,3
```