A Circle and a Square

HackerRank

In this challenge, you must implement part of a raster graphics editor that takes the coordinates of a circle and a square as input and draws them as filled-in shapes on a rectangular canvas.

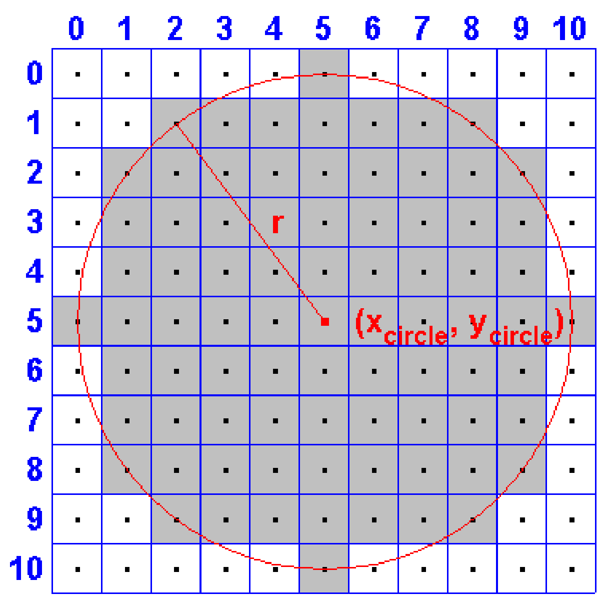
The rectangular canvas consists of uniformly sized square pixels, and is w pixels wide, and h pixels high. Each point on the canvas belongs to a pixel, the intersection of two pixels has zero area, and each pixel is completely contained within the canvas.

The Cartesian coordinate system set up in the following way:

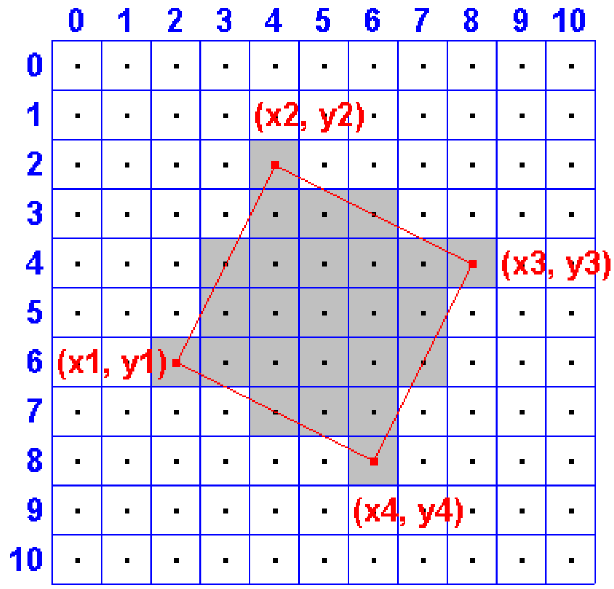
- Point $(0, 0)$ is the center of the top-left pixel of the canvas.
- Point $(w-1, 0)$ is the center of the top-right pixel of the canvas.
- Point $(0, h-1)$ is the center of the bottom-left pixel of the canvas.
- Point $(w-1, h-1)$ is the center of the bottom-right pixel of the canvas.

Thus, all pixel centers have integer coordinates and if the center of a pixel has coordinates (x_c, y_c), then point (x, y) belongs to the pixel if and only if $x \in [x_c - 0.5, x_c + 0.5]$ and $y \in [y_c - 0.5, y_c + 0.5]$. The two shapes should be drawn like so:

- The *circle* is centered at the integer coordinates (x_{circle}, y_{circle}) and has non-negative integer radius r. A pixel should be *black* as a part of the circle if and only if the Euclidean distance from the pixel's center to the center of the circle is *not* greater than r.



- The *square* is defined by the integer coordinates of two of its opposite corners (x_1, y_1) and (x_3, y_3). A pixel should be *black* as a part of the square if and only if its center falls within the square or along its border. The coordinates of different corners of the square do not coincide.



Given h, w, and the definition of the circle and the square, print a raster image of the canvas where each character is either a `.` (denoting a *white* pixel outside of both shapes) or a `#` (denoting a *black* pixel that's part of a shape).

Note: The first pixel of the first line of output should correspond to the top-left corner of the canvas.

Input Format

The first line contains two space-separated integers describing the respective values of w (canvas width) and h (canvas height).
The second line contains three space-separated integers describing the respective values of x_{circle}, y_{circle}, and r defining a circle with radius r centered at (x_{circle}, y_{circle}).
The third line contains four space-separated integers describing the respective values of x_1, y_1, x_3, y_3 defining a square with opposite corners at (x_1, y_1) and (x_3, y_3).

Constraints

- $10 \leq w, h \leq 100$
- $-100 \leq x_{circle}, y_{circle}, x_1, y_1, x_3, y_3 \leq 200$
- $0 \leq r \leq 100$

Output Format

Print h lines where each line contains w characters. Each character must be either a `.` (to denote a white pixel) or a `#` (to denote a black pixel). The first pixel of the first line of output corresponds to the top-left corner of the canvas.

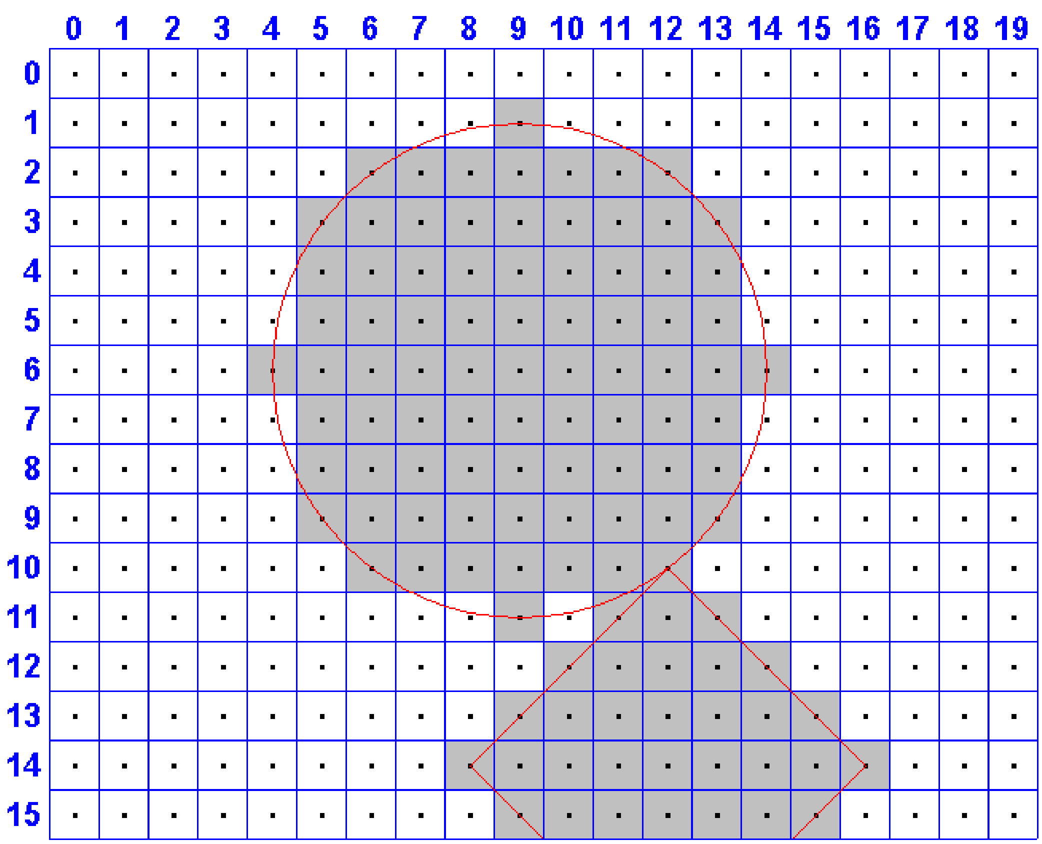
Sample Input 0

```
20 16
9 6 5
16 14 8 14
```

Sample Output 0

```
....................
.........#..........
......#######.......
.....#########......
.....#########......
.....#########......
....###########.....
.....#########......
.....#########......
.....#########......
......#######.......
.........#.###......
..........#####.....
.........#######....
........#########...
.........#######....
```

Explanation 0



The canvas has $h = 16$ rows and $w = 20$ columns. The circle has radius $r = 5$ and is centered at point $(9, 6)$. The square has opposite corners located at points $(16, 14)$ and $(8, 14)$ and, as you can see, is rotated at an angle with its third corner at point $(12, 10)$ (note that its fourth corner is outside the canvas boundary). In addition, the circle and the square overlap at point $(12, 10)$.