2 Line Drawing Algorithms

You may not always see them as such, but lines are continuous with infinite resolution, and are always perfectly precise no matter how you rotate them, scale them, or translate them. Anyone that's enjoyed some pixel art knows that lines at low resolution don't always look like lines. Lines can actually be somewhat challenging to translate to a screen: which pixels do we fill, and which do we leave blank? Enter line drawing algorithms! Line drawing algorithms determine which pixels to fill and leave empty for pretty much any type of line or curve.

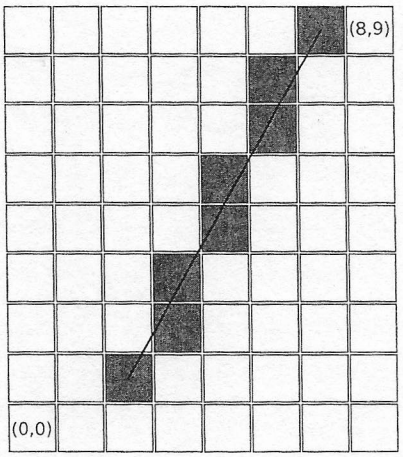

Figure 1: line drawing diagram

Bressenham's line drawing algorithm determines which pixel to fill in by determining the error (the difference) between the center of a pixel and the line; if the error is the right size, then the pixel is filled in. In other words, pick the pixel who's center is closest vertically to the line (or horizontally for a slope greater than 1!).

For this question, given the two endpoints of a straight line segment, determine which pixels to fill with black to create the line.

2.1 Input

Four integers: the x and y coordinates of the line's start-point, and the x and y coordinates of the line's end-point. You may assume that the coordinate values will always greater than zero.

2.2 Output

Integers that represent the integer coordinates of each point to fill with black, resulting in a correctly drawn line.

2.3 Sample IO

Sample Input	Sample Output
5	
0 0 3 4	0 0 1 1 1 2 2 3 3 4
3 4 0 0	3 4 2 3 1 2 1 1 0 0
6 2 9 13	6 2 6 3 7 4 7 5 7 6 7 7 8 8 8 9 8 10 8 11 9 12 9 13
1 1 1 6	1 1 1 2 1 3 1 4 1 5 1 6
3 2 7 9	3 2 4 3 4 4 5 5 5 6 6 7 6 8 7 9