

CS/INFO 3300 // INFO 5100

## Homework 2

Due at 11:59pm Wednesday, September 7

Goals: Get practice using SVG elements.

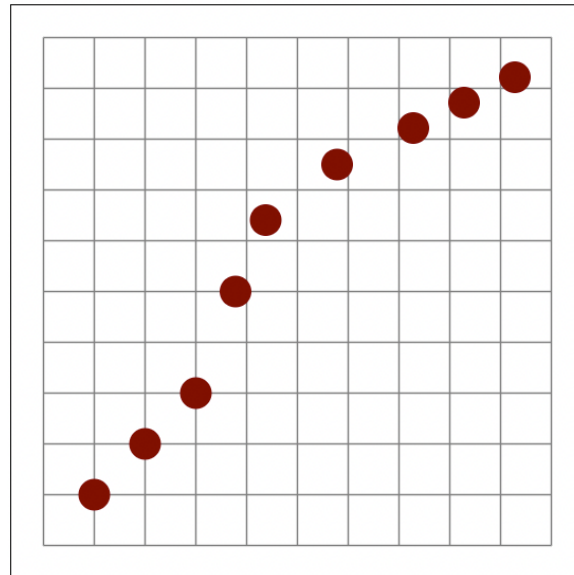
Your work should be in the form of an HTML file called `index.html`. At the top of your `<body>`, please **put your name and netID in a `<h3>` tag**. Include one `<p>` element per problem. Wrap any SVG code for each problem in a `<svg>` element nested within the `<p>` element. For this homework **we will not be using Javascript**.

Since we want you to get more experience working with SVG, we would like you to complete this assignment using **handwritten SVG elements**. Some problems might require numbers like pixel positions which require calculation. Feel free to use whatever tool you prefer in order to compute these, but they should appear in your turn-in homework as literal numbers in the code. You can use a [color picker tool](#) to find good colors, but once again they must appear as literal elements in the file. You **may not** use an SVG authoring tool like Adobe Illustrator. We can tell if you use one, and the time it would take to obfuscate their output is much greater than just doing the assignment yourself.

Create a zip archive containing your HTML file and upload to CMS before the deadline.

(see next page)

1. Create a 360x360 pixel SVG element. Using a `style=""` attribute, give it a 1px solid black border. Reproduce this plot using SVG elements:



The main plot region, excluding axis labels, should be a square 320x320 pixels in size, running from (20,20) to (340, 340). Reserve the remaining pixels on the outside as padding. Data for the points are provided at the bottom of this homework file.

The chart should start with data value [0,0] in the lower left corner (pixel location (20, 340) ) and end with data value [10,10] in the upper right corner (pixel location (340, 20) ).

Create a circle mark for each data point. Circle marks should be reasonably sized and in a dark color of your choosing (the image uses "maroon"). You should calculate the proper pixel positions for the [x, y] coordinates from the data as necessary.

Remember to account for the "padding" pixels when determining positions for points. While we recommend that you hardcode point positions using the 20-340 pixel range, we will also accept submissions that use [SVG group translations](#) to relocate the plot region.

Add horizontal and vertical gridlines for each integer from 0 to 10 in a light grey.

Your result may not look exactly pixel-by-pixel like the figure; we will be grading based on your accuracy in positioning points.

**(next page)**

2. Examine your finished canvas (or the example image). In a `<p>` tag, please identify:

a) the marks used for each row of data

b) the two visual channels used in this visualization

(when answering, please ignore the annotations you added such as gridlines)

3. Use `<line>` and `<rect>` elements to create your own version of a Piet Mondrian painting in a **300x300 SVG** canvas. [Piet Mondrian](#) was an early 20<sup>th</sup> century artist who, as a member of the De Stijl movement, reduced his art to three primary colors and black lines in a series of famous *neoplasticist* works. Here are some examples: [one](#), [two](#), [three](#). In your own neoplastic work, you must include at least **6 lines and 3 rectangles**. `<line>` elements must use a black stroke and `<rect>` elements must be either white, **red, yellow, or blue fill**. `<rect>` elements cannot have a stroke - only use a fill for them. If you want black borders, you must use `<line>` elements. No other colors will be permitted. You may use any additional features you feel would add aesthetic value. If you use a tool to generate coordinates for shapes, please cite that tool. Faithfulness to art history will not be evaluated. We are not grading based on creativity, but obviously poor or incomplete submissions will be penalized. (20pts)

## Data for scatterplot

	<b>X</b>	<b>Y</b>
p1	1.0	1.0
p2	2.0	2.0
p3	3.0	3.0
p4	3.8	5.0
p5	4.4	6.4
p6	5.8	7.5
p7	7.3	8.2
p8	8.3	8.7
p9	9.3	9.2

**Bonus problem** (no extra credit offered – just for fun)

Make a Python program that automatically generates the canvas, lines, labels, and circle marks for problem #2.

Your program should take as input a list of dictionary objects, each with “x” and “y” values. Ideally, you should be able to configure the minimum and maximum values you want to display on the x and y axes.

You can use min, max, and modulo functions to identify how many gridlines to create and identify where your major axis labels (0,5,10...) could go if you want to include them.

Since you know the canvas area is 320x320 and you’ve defined your axis min/max values, you can use some math to figure out where your circles should go.

Afterwards, use string composition to output some valid HTML.

Use this to check your answer to #2.

...this might also help:

```
data = [{"x":1.0 , "y":1.0},
        {"x":2.0 , "y":2.0},
        {"x":3.0 , "y":3.0},
        {"x":3.8 , "y":5.0},
        {"x":4.4 , "y":6.4},
        {"x":5.8 , "y":7.5},
        {"x":7.3 , "y":8.2},
        {"x":8.3 , "y":8.7},
        {"x":9.3 , "y":9.2}]
```