Lab 4

CST 205

If someone told you to create a grayscale Mona Lisa using the following text, would they be crazy?

```
132 128 126 123 137 129 130 145 158 170 172 161 153 158 162 172 159 152;
139 136 127 125 129 134 143 147 150 146 157 157 158 166 171 163 154 144;
144 135 125 119 124 135 121 62 29 16 20 47 89 151 162 158 152 137;
146 132 125 125 132 89 17 19 11 8 6 9 17 38 134 164 155 143;
142 130 124 130 119 15 46 82 54 25 6 6 11 17 33 155 173 156;
134 132 138 148 47 92 208 227 181 111 33 9 6 14 16 70 180 178;
151 139 158 117 22 162 242 248 225 153 62 19 8 8 11 13 159 152;
153 135 157 46 39 174 207 210 205 136 89 52 17 7 6 6 70 108;
167 168 128 17 63 169 196 211 168 137 121 88 21 9 7 5 34 57;
166 170 93 16 34 63 77 140 28 48 31 25 17 10 9 8 22 36;
136 111 83 15 48 69 57 124 55 86 52 112 34 11 9 6 15 30;
49 39 46 11 83 174 150 128 103 199 194 108 23 12 12 10 14 34;
26 24 18 14 53 175 153 134 98 172 146 59 13 14 13 12 12 46;
21 16 11 14 21 110 126 47 62 142 85 33 10 13 13 11 11 15;
17 14 10 11 11 69 102 42 39 74 71 28 9 13 12 12 11 18;
18 19 11 12 8 43 126 69 49 77 46 17 7 14 12 11 12 19;
24 30 17 11 12 6 73 165 79 37 15 12 10 12 13 10 10 16;
24 40 18 9 9 2 2 23 16 10 9 10 10 11 9 8 6 10;
43 40 25 6 10 2 0 6 20 8 10 16 18 10 4 3 5 7;
39 34 23 5 7 3 2 6 77 39 25 31 36 11 2 2 5 2;
17 16 9 4 6 5 6 36 85 82 68 75 72 27 5 7 8 0;
4 8 5 6 8 15 65 127 135 108 120 131 101 47 6 11 7 4;
2 9 6 6 7 74 144 170 175 149 162 153 110 48 11 12 3 5;
11 9 3 7 21 127 176 190 169 166 182 158 118 44 10 11 2 5;
8 0 5 23 63 162 185 191 186 181 188 156 117 38 11 12 25 33;
3 5 6 64 147 182 173 190 221 212 205 181 110 33 19 42 57 50;
5 3 7 45 160 190 149 200 253 255 239 210 115 46 30 25 9 5;
9 4 10 16 24 63 93 187 223 237 209 124 36 17 4 3 2 1;
7 8 13 8 9 12 17 19 26 41 42 24 11 5 0 1 7 4;
```

Recall that for a grayscale image each pixel is stored as an 8-bit integer. As a result, possible values range from 0 to 255. 0 is black, 255 is white, values in between are shades of gray.

Deliverable

Submit your source code (with explanatory comments).

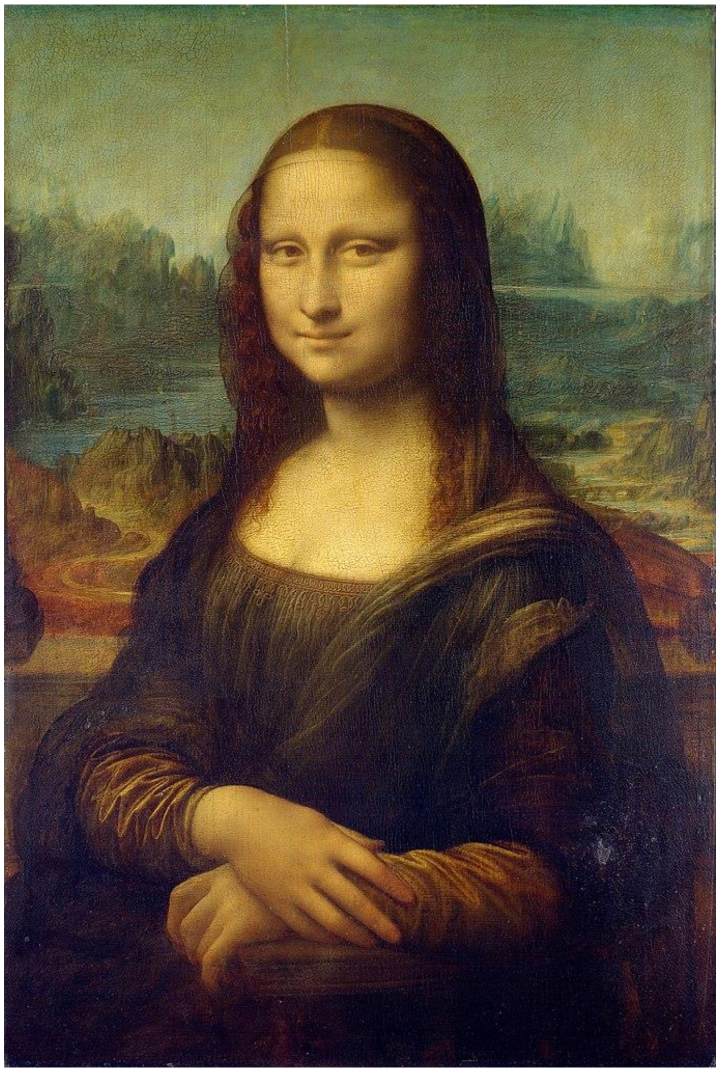
(Go to the next page for hints. Try it first without any hints or help.)

Hints and help

When given data like this, we first need to make sense of it. We might have to make a few assumptions and try a few things out before we find the solution. For the data above, we see several lines each ending in a semi-colon. Each row has 18 numbers. There are 29 rows. Each value represents a grayscale pixel at its position on the canvas, so we have a total of 18x29 = 522 pixels. (The Mona Lisa image below for example has 703,488 pixels for comparison.)

Here is a spreadsheet with the numbers (if you find this format helpful):
https://docs.google.com/spreadsheets/d/1r-KUvh9Ul4_uJyPOAIQsr3anetkbA2JsdqgVWvDC2sA/edit?usp=sharing

Here is the Mona Lisa (in color), just to get an idea of the image composition:



If we look at the *really small* values (say 0-9) they seem to correspond to the darker areas of the image, and the really big values (say 200-255) seem to correspond to the lighter areas. (The data provided is from a cropped image, so we can't draw *too* much of a comparison here.)

We can certainly manually tweak our numbers to get them in a more useful format, but we can use Python to make our lives easier:

To store multi-line strings in Python we want to surround our string in three single quotes: `''' ... long multi-line string ... '''`

We want to assign our long, multi-line string to a variable, say `mona_string`.

Python provides a variety of methods for splitting strings, but for our purposes, we can use Python's `splitlines()` method.

`mona_lists = mona_string.splitlines()` will store each line in a list (as a string).

We can loop through our lists and get our data in better shape by removing the trailling semi-colon and converting our strings to lists. (Later in the course we will learn about **list comprehension**, which is generally considered a better way of doing this.)

One way to write this for loop is to use `enumerate` which gives us a counter for free. Within this loop we can use Python's `rstrip()` method to remove a rightmost character. We can also then split our lines into a list of strings.

So, line 1 will go from:

```
'132 128 126 123 137 129 130 145 158 170 172 161 153 158 162 172 159 152;'
```

to:

```
['132', '128', '126', '123', '137', '129', '130', '145', '158', '170', '172', '161',
'153', '158', '162', '172', '159', '152']
```

Here's one way to do the loop:

```
for count, mona_line in enumerate(mona_lists):
  mona_line_clean = mona_line.rstrip(';')
  mona_lists[count] = mona_line_clean.split()
```

Believe it or not, at this point we are able to construct a loop (similar to the loop in our Digital Image Filtering lecture), to loop through every pixel in the image and assign a value to the pixel. On slide 13 from the our Digital Image Filtering lecture, we looped through an image and used Pillow's `getpixel()` method. For Mona Lisa, we will use Pillow's `putpixel()` method, since we want to place image information at each pixel.

However, just as Leonardo da Vinci needed a blank canvas before he began to paint the Mona Lisa, we similarly need a blank canvas of the correct size. Here's how we do this using Pillow:

```
width = 18
height = 29
my_mona = Image.new('L', (width, height))
```

We use `L` since this is a grayscale image. For Mona Lisa, the width refers to the number of values per row and the height refers to the number of columns.

Now, we are ready to do our nested loop. Conceptually, this may be a little confusing as each of our lists corresponds to a row of pixels. Our loop, however, goes down columns of data, so we want to transpose using `mona_lists[y][x]` instead of `mona_lists[x][y]`. (Try a few examples by hand if you need more convincing.)

We also need to provide Pillow with an integer, not a string, so we need to cast our data.

```
for x in range(width):
  for y in range(height):
    my_mona.putpixel((x,y), int(mona_lists[y][x]))
```

We need to save our image:

```
my_mona.save('mona.png')
```

Last step, is to run your program, sit back and enjoy your masterpiece (after zooming in a bit).