Information and Communication Technologies — S6 — Work n°21

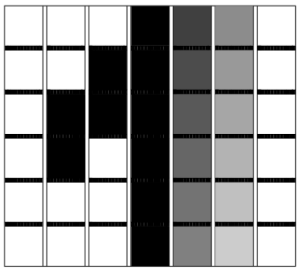
Digital images — Part 2: Handling (some) image formats

1 Black and white images: the portable graymap format (PGM)

One format to manipulate black and white images is the portable graymap format (PGM) file, see for example https://en.wikipedia.org/wiki/Netpbm#File_formats. In this format, 0 is black and 255 is white. I here recall the image from last week. Its PGM encoding is also shown in Figure 1, and you can download it at http://www.barsamian.am/2020-2021/S6ICT/TP21_Image1.pgm.

⚠ When using this file format, no line can exceed 70 characters (if you somehow need more, just open a new line). As in Python, lines that begin with a # are ignored (they are just comments).

$$image = \begin{pmatrix} 255 & 255 & 255 & 0 & 65 & 140 & 255 \\ 255 & 255 & 0 & 0 & 77 & 153 & 255 \\ 255 & 0 & 0 & 0 & 97 & 166 & 255 \\ 255 & 0 & 255 & 0 & 102 & 179 & 255 \\ 255 & 255 & 255 & 0 & 115 & 191 & 255 \\ 255 & 255 & 255 & 0 & 128 & 204 & 255 \end{pmatrix} \text{ represents the image}$$



```
P2
# "P2" means that this is a PGM file, stored in ASCII form
# "7 6" means that this is an image of size 7 x 6 (width x height)
# "255" means that the maximum value for a pixel is 255 (for white)
7 6
255
255 255 255 0  65 140 255
255 255   0 0  77 153 255
255   0   0 0  97 166 255
255   0 255 0 102 179 255
255 255 255 0 115 191 255
255 255 255 0 128 204 255
```

Figure 1: A sample image in gray scale, stored in the PGM file format.

We will first work on our school's logo. Start by downloading the following files:

http://www.barsamian.am/2020-2021/S6ICT/TP21_Logo_EEB1.pgm
http://www.barsamian.am/2020-2021/S6ICT/TP21_Images_logo.py

1. Write the function `invert` that inverts the grayscales of the image. You can safely assert that the maximum value of the grayscale is 255.

   First, from a given two-dimensional array containing the grayscales, you must extract the width and the height of it. The two-dimensional array is given line by line (hence, the length of the array is the number of lines — the height — and the length of any line is the number of columns — the width). See for instance in the function `store_image` how this information is extracted.

   Then, you must create a new two-dimensional image. You can either create a fresh new array as in Listing 1, or make a copy of the old one, as in Listing 2.

```python
new_image = [[0 for y in range(width)] for x in range(height)]
```

Listing 1: Sample code that creates a new 2d array.

```python
import copy
new_image = copy.deepcopy(image)
```

Listing 2: Sample code that creates a copy of a 2d array.

Finally, for each value v, the new value is the inverse, $255 - v$.

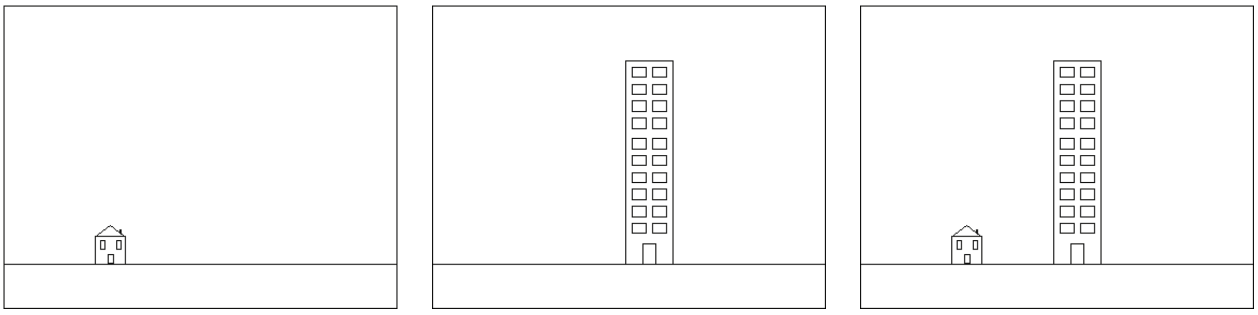
2. The implementation given should now have created a new image `TP21_Inverted_logo.pgm`. Open this image in any image manipulation program to check your result.

We will now work on multiple pgm images at once. Start by downloading the following files:

http://www.barsamian.am/2020-2021/S6ICT/TP21_House.pgm
http://www.barsamian.am/2020-2021/S6ICT/TP21_Skyscraper.pgm
http://www.barsamian.am/2020-2021/S6ICT/TP21_Images_merge.py

3. Implement the function `merge` that takes as input two arrays representing pgm images and outputs the merging of the two arrays. The implementation given should now have created a new image `TP21_Merged.pgm`. Open this image in any image manipulation program to check your result. The result should be the one on the right:

BONUS How would you modify your program to handle the case where the widths, heights, and/or maximum values are not the same?



2 Colored images: the portable pixmap format (PPM)

We will now work on ppm images at once. Start by downloading the following files:

http://www.barsamian.am/2020-2021/S6ICT/TP21_Joconde_Original.ppm
http://www.barsamian.am/2020-2021/S6ICT/TP21_Images_colors.py

4. Write the function `invert` that inverts the red, blue and green values of the image. You can safely assert that the maximum value of the values is 255. The result should be the one on the middle.

5. Write the function `to_gray` that converts each pixel (r, g, b) to a gray pixel. A gray pixel can be defined by using all three values equal. We will here choose, for each pixel, the average of the three values r, g, b. You can safely assert that the maximum value of the values is 255. The result should be the one on the right.