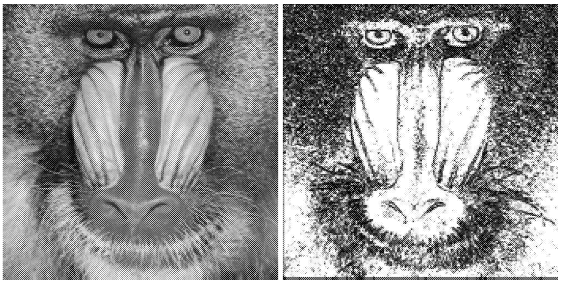
Data Structures and Programming
Spring 2019
Programming Assignment #2
Due: June 14, 2019.

The purpose of this assignment is to implement a program that reads in an image in PGM format, perform edge detection on the input, and output the resulting image. The following figure shows an example input image (left) and the corresponding output (right).



PGM File format PGM, or portable graymap format, is an image format for storing uncompressed grayscale image. There are several variations of PGM file format (man pgm for details.), but for the purpose of this assignment, we assume the following simplified format. Each PGM file consists of the following:

- A magic number, which is the two characters "P2".
- Whitespace (blanks, TABs, CRs, LFs).
- A width, w, formatted as ASCII characters in decimal.
- Whitespace.
- A height, h, again in ASCII decimal.
- Whitespace.
- The maximum gray value (maxval), again in ASCII decimal. You can assume this to be 255.
- Newline or other single whitespace character.
- A raster of $w \times h$ gray values, proceeding through the image in normal English reading order (i.e., left to right, top to bottom). Each gray value is a number from 0 through 255, with 0 being black and 255 being white.
- Lines that begin with a #, before the maxval line, are comments and are ignored.

Your program must be able to ignore comments, and handle images of arbitrary size. If the magic number P2 cannot be found at the beginning of the image, your program should output an error and quit. Otherwise, you can assume that the rest of the file is a valid PGM file. Your

```
P2
# feep.ascii.pgm
24 7
15
0 0  0  0  0  0  0  0  0 0  0  0  0  0  0  0  0 0  0  0  0  0  0  0  0
0 3  3  3  3  0  0  7  7 7  7  0  0 11 11 11 11 0  0 15 15 15 15  0
0 3  0  0  0  0  0  7  0 0  0  0  0 11  0  0  0 0  0 15  0  0 15  0
0 3  3  3  0  0  0  7  7 7  0  0  0 11 11 11  0 0  0 15 15 15 15  0
0 3  0  0  0  0  0  7  0 0  0  0  0 11  0  0  0 0  0 15  0  0  0  0
0 3  0  0  0  0  0  7  7 7  7  0  0 11 11 11 11 0  0 15  0  0  0  0
0 0  0  0  0  0  0  0  0 0  0  0  0  0  0  0  0 0  0  0  0  0  0  0  0
```

program should parse the header and read the pixel (raster) data into a dynamically allocated 2D array.

Edge Detection

We now describe how to perform edge detection and compute the output image. Refer to the following fugure. Given a pixel on the output image e, we compute two intermediate values x and y using the neighboring pixels of e from the input as follows:

$$x = (c + 2f + i) - (a + 2d + g)$$

$$y = (g + 2h + i) - (a + 2b + c)$$

a	d	g
b	e	h
c	f	i

Figure 1: To compute the value of output e, we make use the values of the surrounding pixels (a, b, c, d, f, g, h and i) from the input image.

The value of the output pixel at location e, is then computed using

- $e = 0$ if $\sqrt{x^2 + y^2} \geq \epsilon$;
- $e = 255$, otherwise.

where ϵ is a threshold value which you must define as a constant using #define (or equivalent statement in your chosen programming language. A good value for ϵ is 128. You can try different values for ϵ. The above description does not take care the cases where e lies at the border of the image. For simplicity, you can simply set of the value of the border pixel to 255.

First, you must process the files given on the website and produce outputs. Second, if your program is called, a file name ("test.pgm") is passed to your program as command line

argument, your program should read the input PGM image from the given file. In case more than one command line argument is passed to your program, your program should simply remind the user the usage of the program and quit.

More will be said about what you should turn in later.

BONUS

- Design and implement other template-based edge detection algorithms.