CSCD 240
Lab 11

Problems Description

In this project, we manipulate PGM (Portable Gray Map) images. PGM files are pure ASCII text files, with header information and intensity value for each pixel in the image file. If you have an image ballon.pgm, you can use command 'less ballon.pgm' to explore its format. Surely you can install an image viewer to visualize the image with you naked eye.

On windows, you can use **Irfanviw**, http://www.irfanview.com
On a Mac machine, you can download **ToyViwer** in your apple store for free.

The detailed format for PGM file can be found here, and you can download some sample PGM files **http://people.sc.fsu.edu/~jburkardt/data/pgma/pgma.html**

The following is an example PGM file named smallFile.pgm

```
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 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
```

The image header consists of the first 4 lines.

1. The first line 'P2', is a magic number to tell the image viewer that the file is a ASCII pgm file.
2. The second line starts with #, is a line of comment.
3. The numbers in the third line, means this image has 24 COLUMNS, and 7 ROWs of pixels in it.
4. The fourth line means the maximal intensive value in the image is 15. Each pixel has intensive value. **The bigger intensive value is, the brighter or whiter at that location is in the image. The intensive value 0 means total black in the image.**

Intensity values for all pixels are listed after the file header, starting with line 5. These intensity values could be considered as a 2D array, with the row index and column index identifying each point or pixel at a particular location.

What you need to do?

1) Implement the function that read in the image as dynamic 2D array from a text file using FILE *.
2) Implement the function to paint a black dot (circle) in the image; you have to parameterize the center point and the radius of the circle you will draw.
3) Implement the function that paints a black edge frame in the image, you have to parameterize the width of the edge you will paint.
4) Implement the function that writes back to a new image file that contains your painting. This will occur using FILE *.
5) Make sure you clean up all dynamically allocated memory
6) You will write code to parse the command-line arguments.
   Usage:
   -e edgeWidth oldImageFile newImageFile
   -c circleCenterRow circleCenterCol radius oldImageFile newImageFile

7) If the number of command line argument is not expected, your program are required to prompt for the information
8) Command line arguments will be similar to the following

   ./programName -e edgeWidth originalImage newImageFile

   OR

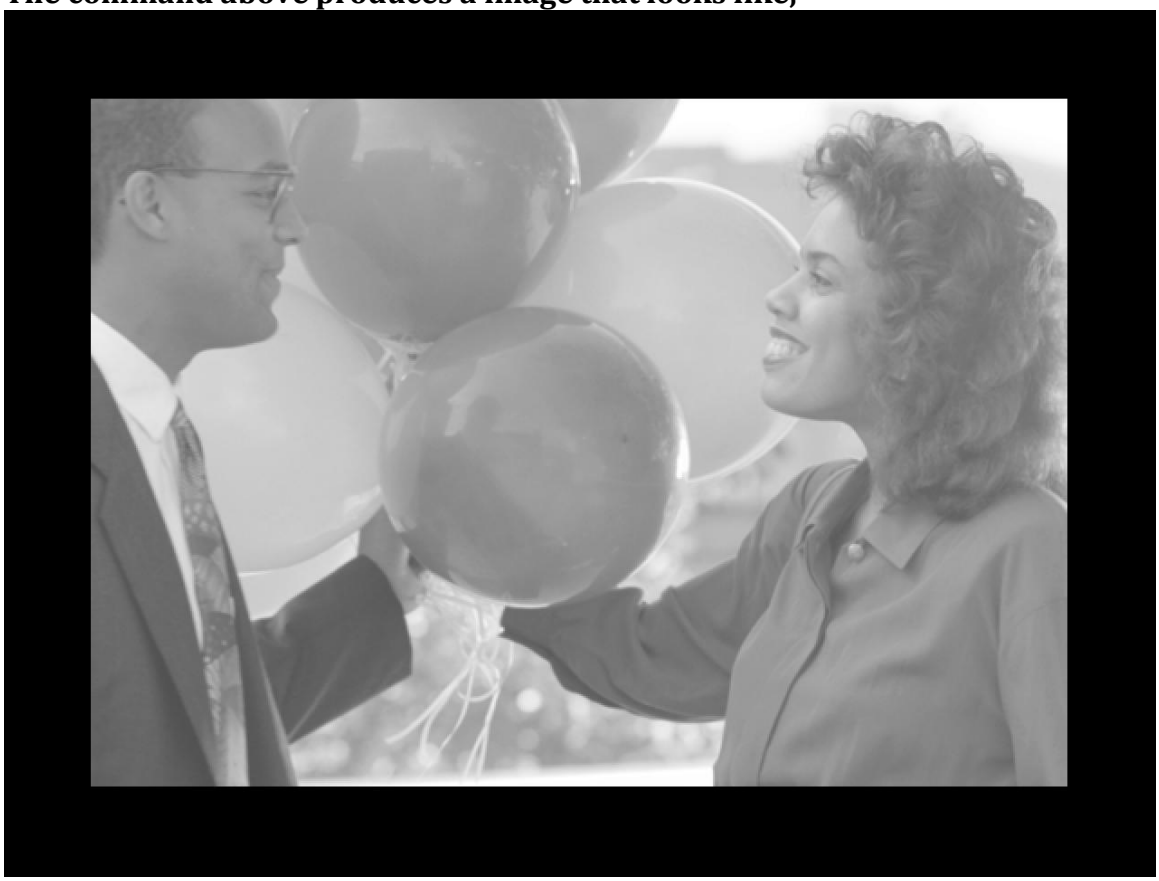
   ./programName -c circleCenterRow circleCenterCol radius originalImage newImageFile

Test cases with the provided image

./myPaint -c 470 355 100 balloons.ascii.pgm balloons_c100_4.pgm
Your program yields an image looks like:

./myPaint -c 228 285 75 balloons.ascii.pgm balloons_c75_5.pgm
The command above yields an image,

./a.out -e 50 balloons.ascii.pgm balloons_e50_2.pgm
The command above produces a image that looks like,



TO TURN IN

A **zip** file that contains: - **You better know the naming scheme by now**

- All C files, you will use a 3 file format
- All testing image files
- All testing output files
- A makefile to compile your code – target will be named lab11
- A valgrind run to show your are leak free

NOTE: only compile your code in the makefile do NOT run your code