**CSCD 240 Homework 34 (Project ) (Total 100 points)**

**Simple Image Processing using C pointer to pointers, string operations,**

**type conversion, math library, command-line parsing and file I/O**

**Description**

In this project, we manipulate PGM images (Portable Gray Map). A PGM file could be stored in pure ASCII text files, with header information and intensity (brightness) value for each pixel in an image file. If you have an image named ballon.pgm, you can use command ***less ballon.pgm*** to explore its format. Of course, you can install an image viewer to visualize the image. On windows, you can use **Irfanviw**, and download is available at <http://www.irfanview.com>

On a Mac machine, you can download **ToyViwer** in your **apple store for free**.

**The detailed format description for a PGM file can be found here.** (Also you can download more sample PGM files, besides one included in this project package.)

[**http://people.sc.fsu.edu/~jburkardt/data/pgma/pgma.html**](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 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

**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. **Note: column goes first!**
4. The fourth line means the maximum intensity value in the image is 15. Each pixel has intensity value. **The bigger intensity value is, the brighter or whiter at that location is in the image. The intensity value 0 means a total black color in the image.**

Intensity values of all pixels are listed following the file header, starting with 5th line. These intensity values could be considered as a 2D array, with the **row index** and **column index** specifying a particular location on the image.

**What you should do?**

**Please carefully read the comments on top of each function declared in pgmUtility.h file.**

1. In a **pgmUtility.c** file, please read the function that reads in the image **(actually a text file)** using file I/O. The function has been declared already in pgmUtility.h file and has been implemented in the pgmUtility.c file. Please read and understand this function.

int \*\* pgmRead( char \*\*header, int \*numRows, int \*numCols, FILE \*in );

1. In the **pgmUtility.c** file, please implement the function that paints a black dot (circle) in the image, you have to parameterize the center point and the radius of the circle you will draw. The function has been declared already in pgmUtility.h file.

int pgmDrawCircle( int \*\*pixels, int numRows, int numCols, int centerRow,

int centerCol, int radius, char \*\*header );

1. In the **pgmUtility.c** file, please implement the function that paints a black edge frame in the image, you have to parameterize the width of the edge you will paint. The function has been declared already in pgmUtility.h file.

int pgmDrawEdge( int \*\*pixels, int numRows, int numCols, int edgeWidth, char \*\*header );

1. Note that after you paint the edge or the black dot in an image, you have to update the header to reflect the new maximum intensity value in the header (specifically, in the last line of image header). NOTE: this operation might render the new image brighter when you compare with the original image.
2. In the **pgmUtility.c** file, please read the function that writes back to a **new** image file that contains your painting using file I/O. The function has been declared already in pgmUtility.h file and has been implemented in pgmUtility.c file. Please read and understand this function.

int pgmWrite( const char \*\*header, const int \*\*pixels, int numRows, int numCols, FILE \*out );

1. You are required to implement the functions that are declared in the \*.h file, you CANNOT change the signatures (parameter list and return type) of these functions. You have to place the implementation of the functions into a separate **pgmUtility.c** file, and you have to jointly compile multiple source files in this project. **Please carefully read the comments on top of each function declared in pgmUtility.h file.**
2. You are required to deallocate all memories you dynamically allocated in program. Please make sure you do it at a proper time and at proper location in your program, where you know these memories are no longer useful. In addition, you are required to check your memory deallocation **by running your program with valgrind command on cslinux machine**. **Please create a pdf file that shows the valgrind memory-check result.**
3. Add a simple makefile to compile your source code into an executable hw34.
4. I have provided a main() function in a main.c file. In main(), you call the above-mentioned function(s) in order to **output a new image file on hard disk**. You may freely change the provided main() function to fulfill the requirements.
5. The sample runs provided in the next section are explaining the same information here below. When run your program, you have to enter the input file name and output file name from **stdin**, also you have to choose what object you like to draw on an input image, then followed by either the Circle Center, Radius or Edge Width on the **standard input**. Please refer to the following section for sample runs.
6. Note that your program will process a single input image per each execution. You are **not** required to display the output image in a GUI window during execution. Rather, your program writes back the image into a file on the hard disk, then, you can open it with an image viewer tool.

**Test cases with the provided image**

Sample run (1):

./hw34

**Please enter the input file name, including its file extension:**balloons.ascii.pgm

**Please enter the output file name, including its file extension:**balloons.out.c1.pgm

**------------------Menu--------------:**

**1--draw an edge in a picture.**

**2--draw a circle in a picture.**

**------------------------------------:**

**Please choose a menu item:** 2

**Please enter the circle center's row index and column index:** 470 355

**Please enter the radius of the circle:** 100

**Output image balloons.out.c1.pgm has been generated with a circle on it.**

**Your program yields an image looks like:**

****

Sample Run (2)

./hw34

**Please enter the input file name, including its file extension:**balloons.ascii.pgm

**Please enter the output file name, including its file extension:**balloons.out.c2.pgm

**------------------Menu--------------:**

**1--draw an edge in a picture.**

**2--draw a circle in a picture.**

**------------------------------------:**

**Please choose a menu item:** 2

**Please enter the circle center's row index and column index:** 228 285

**Please enter the radius of the circle:** 75

**Output image balloons.out.c2.pgm has been generated with a circle on it.**

**Your program yields an image looks like:**

****

**Sample Run (3):**

**./hw34**

**Please enter the input file name, including its file extension:**balloons.ascii.pgm

**Please enter the output file name, including its file extension:**balloons.out.e1.pgm

**------------------Menu--------------:**

**1--draw an edge in a picture.**

**2--draw a circle in a picture.**

**------------------------------------:**

**Please choose a menu item:** 1

**Please enter the edge width measured in number of pixels:** 50

**Output image balloons.out.e1.pgm has been generated with an edge frame on it.**

**Your program yields an image looks like:**

****

**To turn in:**

Please wrap up all source code, make file, your pdf file that shows the valgrind check result and all included input images into a single zip file, name it as lastname + firstinitial + h34.zip. If you are Will Smith, your zip file is named as smithwh34.zip.

**If you did the bonus points part, please clearly state that on the top of your pdf file originally used for showing valgrind check result.**

Turn in your single zip file on Canvas CSCD240 -> Assignments->hw34->submit

**If your code shows a compile error or a segmentation fault, you will get zero credit. If your file is corrupted, you get a zero credit.**

**All description below is for Optional BONUS POINTS (total 35 points)**

To earn these bonus points, when you run your program, you have to pass in file names and the Circle Center, Radius or Edge Width as **command line arguments**. In your main function, you have to write code to parse the command-line arguments. Also, if the number of command line argument is not expected, your program are required to show a message as below:

***Usage:***

***./hw34 -e edgeWidth oldImageFile newImageFile***

***./hw34 -c circleCenterRow circleCenterCol radius oldImageFile newImageFile***

You have to run your program using command with this synopsis:

***./hw34 –e edgeWidth originalImage newImageFile***

in order to paint an edge of width of ***edgeWidth*** in the image of ***originalImage***

***./hw34 –c circleCenterRow circleCenterCol radius originalImage newImageFile***

in order to paint an big round dot on the image with center at (***circleCenterRow, circleCenterCol***) and radius of ***radius***.

When an user inputs wrong number of argument in **command line** or in a wrong format, your program **should not crash**, instead showing the Usage message above.

If your program could handle drawing a circle and an edge at the same time in one command, you get another **10** **bonus points** on top of 125.

In this case, your command line should look like the following; otherwise you cannot get the bonus.

***./hw34 -ce circleCenterRow circleCenterCol radius edgeWidth originalImage newImageFile***

***AND***

***./hw34 -c -e circleCenterRow circleCenterCol radius edgeWidth originalImage newImageFile***

**Bonus Section Sample Runs:**

**./hw34 -c 470 355 100 ./balloons.ascii.pgm balloons\_c100\_4.pgm**

**Your program yields an image looks like:**

****

**./hw34 -c 228 285 75 ./balloons.ascii.pgm balloons\_c75\_5.pgm**

**The command above yields an image,**

****

**./hw34 -e 50 ./balloons.ascii.pgm balloons\_e50\_2.pgm**

**The command above produces an image that looks like,**

****

**If input wrong command line parameters:**

./hw34 -e 50 300 ./balloons.ascii.pgm

***Usage:***

***-e edgeWidth oldImageFile newImageFile***

***-c circleCenterRow circleCenterCol radius oldImageFile newImageFile***

./hw34 -e ab ./balloons.ascii.pgm

***Usage:***

***-e edgeWidth oldImageFile newImageFile***

***-c circleCenterRow circleCenterCol radius oldImageFile newImageFile***

./hw34 -e ./balloons.ascii.pgm

***Usage:***

***-e edgeWidth oldImageFile newImageFile***

***-c circleCenterRow circleCenterCol radius oldImageFile newImageFile***

./hw34 -c 470 355 ./balloons.ascii.pgm

***Usage:***

***-e edgeWidth oldImageFile newImageFile***

***-c circleCenterRow circleCenterCol radius oldImageFile newImageFile***

./hw34 -c 470 355 50 60 ./balloons.ascii.pgm

***Usage:***

***-e edgeWidth oldImageFile newImageFile***

***-c circleCenterRow circleCenterCol radius oldImageFile newImageFile***

./hw34 -c 470 90bc ./balloons.ascii.pgm

***Usage:***

***-e edgeWidth oldImageFile newImageFile***

***-c circleCenterRow circleCenterCol radius oldImageFile newImageFile***

**The original image before your processing looks like:**

