**ASSIGNMENT OVERVIEW:**

This assignment will build on the knowledge you now have of PPM files. You are going to write a program that will read from a file the following:

1. Header information for a ppm file.
2. Three values that will represent the data needed to define a circle.
3. Three values that will represent the color for the circle.

Using these values, you will create a ppm image that has one circle as defined by the input file. The color of the circle is also defined by the input file. The background color for the image will be your choice.

**ACADEMIC INTEGRITY:**

This is an individual assignment. If you need help with this assignment, you can seek help from any of the lab TA’s. **You are not allowed to work with any other student. You are not allowed to get code from any outside source. Warning: I plan run this program through a similarity program to determine any academic integrity violations. Please do not copy, use AI, etc.**

**Files you are required to have:**

**driver.c**

This file is where main will live. This file should have minimal amount of code.

1. Open the files
2. Use assert to check if the files opened successfully
3. Call a function to read the information from the input file
4. Call a function to write the header to the output file
5. Call a function that will create the circle
6. Close the files

**ppm.h**

This file should have function prototypes related to ppm images. It should also define any structs related to ppms.

1. Struct to represent the pixels.
2. Struct to represent the header.

Functions:

1. A function to write the header to the the output file.
2. A function to write one pixel to the output file.

There should be no code in the header files. You are required to use header guards in the files.

**ppm.c**

This file has the implementation of the functions in ppm.h.

**shape.h**

This file will have function protypes related to shapes. It should also contain at least 2 structs

1. Struct to represent a point: x and y values. These should be integers not floats.
2. Struct to represent a circle. This should contain the following:
   1. a point that represents the center of a circle
   2. an integer that represents the radius for the circle
   3. a pixel that represents the color values for the circle

**Functions:**

1. A function that will read from the input file that contains the information for the header and the data that describes the circle: the center point, radius, circle color.
2. A function that will loop through each row and column (each point (pixel) location) of the image checking if the point (pixel) in the image is within the defined circle. If the pixel is within the circle, the pixel color will be set to the circles color and written to the output ppm image. If it is not within the circle, the color will be the default color defined by you and written to the output ppm image.

**shape.c**

This file should implement all functions specific to the shape.

You are required to use header guards in each of the .h files. Points will be deducted if header guards are not used. When designing your functions, they must be small and do specific task. Large clunky functions will result in a reduction of points.

**Hint:** I suggest number 2 above be broken into 2 functions. One to loop through the pixels and for each pixel call another function to check if the pixel is within the circle returning true or false. If true the color is the color of the circle, if false the pixel is the color of the background which you will define.

**Input File Info**

I will provide an input file and a ppm file that represents You should create another input file to test your program.

**Circle info:**

A circle has one point (x and y) that represents the center of the circle. It has an integer value for the radius.

The input file will have the following information in this order:

P6 800 800 255

400 400 100

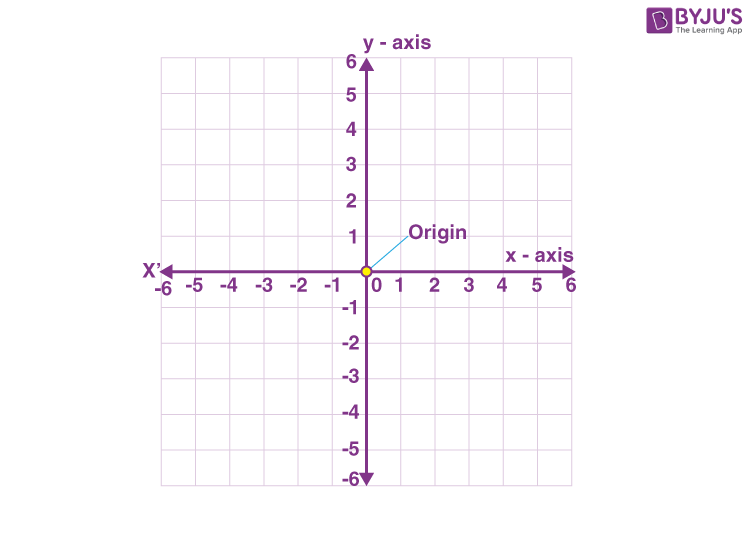
255 0 0

Explanation of the input file:

The first line is the header information of the ppm image you are going to create.

The second line of data represents:

1. The point values for **x and y** that represents the **center of the circle**. Remember x will represents width and y represents height. See graph below.
2. The radius of the Circle.



The third line in the input file represents the RGB color values for the Circle.

With the information contained in the input file you are going to create a ppm image that has one circle, like the following. You are allowed to define the color of the background. The color, size, and placement of the circle is determined by the input file.

A picture containing icon

Description automatically generated

Given the size of the image you are going to determine the color of each pixel in your image. This will be done by checking each pixel location in the image to determine if the location is within the circle’s parameters. If it is, the pixel color will be the color defined in the input file. If it is not within the circle’s parameters, the color will be a predefined color of your choice.

**Circle Formula**

A circle is defined by a point that represents the center of the circle and an integer that represents the radius of the circle. With this information and the point that represents a pixel on a plane, determining if the point is within the circle is a relatively simple formula.

**If the square root of ((x – center x)2  + (y – center y)2) is less than the radius, then the point is within the circle.**

**Makefile:**

You are expected to provide a makefile that has a make run. We should be able to type make run and your program compile and run. The file that has the circle information defined, MUST be named input.txt the output file be output.ppm. I will develop and provide to the TA’s multiple input files to test your program. **Do not change the information order of the input file. We will create our own input file to test with.**

**FORMATTING:**

You will need to add a header to each of your files similar to the following:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*Your name

\*CPSC 2310 <your Lab Section>

\*Your email

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Your program should compile with no warnings and no errors. If your program does not compile the highest grade you can get for the assignment will be 20. If your program compiles but has warnings, there could be a deduction up to 20 points.

* Your code should be well documented. (comments)
* **You must have comments in the header files (.h) See instructions below**
* There should be no lines of code longer than 80 characters.
* You should use proper and consistent indention.

Here are some guide lines for documenting the code in your assignment.

Before each function you are required to have a detailed description of what the overall function does. You should explain what each parameter is and what it is used for.

/\* Parameters: img - image\_t pointer array holding the image data for  
 \*                   each of the input files  
 \* Return:     output - image\_t struct containing output image data  
 \* This function averages every pixels rbg values from each of the   
 \* input images and puts those averages into a single output image  
 \*/

Also, if you include comments in the body of the function (and you should) they should be placed above the line of code not beside the code.

Example:

Bad

if(something) //This is a comment

{

do something;

}

Good

//This is a comment

if(something)

{

do something;

}

**HANDIN:**

Use handin.cs.clemson.edu to submit your files.

Things to do prior to handing in your files.

1. **Test your program on the SoC’s cerf15**.
2. Tar zip your files naming the tarred file: <username>\_Lab6.tar.gz. Make sure your tarred files are not nested in one or more folders. When I untar your file I should immediately be able to type make or make run and the program run. If you have your files nested and the makefile does not run due to the nested files points will be deducted.
3. Make sure you check and double check the files you handin. If you are missing any file, image, etc., you will get an automatic 0. It is your responsibility to make sure your files are correct and have not been corrupt during the handin process. You can check this by reviewing what you actually turned in through the handin page. This will take time so don’t wait until the last minute to hand in this project.