<u>Warhol Image Generator</u>
<u>CS 5001 Intensive Foundations of Computer Science</u>

In this assignment, students will put together several concepts that have already been encountered to create a useful and visual application. Make sure to pay special attention in recitation. I major component will be developed there and must be understood before the rest of the assignment can be completed.

1. Goals:

1. Explore the use of for loops in a meaningful way
2. Reinforce previous concepts of command line usage, file IO, and graphics
3. Get practice in a more complex application

2. In Recitation:

The primary purpose of this recitation is to give you the tools you need to complete an image-manipulation project. We will be importing and using the John Zelle's graphics package. In industry you will often be working with other people's existing libraries. I'd like you to see an example of this with the included graphics package. At different times, you'll need to refer to the existing code to figure out how to implement the steps below. Anytime you are using a function from the GrahpicsPlus file take a second to go look up what it does, what it expects as input, and what it should return. As you will see, code documentation is very important.

In the end, this should give you the skills to complete this week's lab assignment.

<u>Objective 1 : Setup</u>

Make a project6 folder for this week. Put a copy of the grahpicsPlus.py file into this folder.

<u>graphicsPlus.py</u>

You'll be creating three more files: show.py, filter.py, and warhol.py.

Download one or more of the following images. Note that they are ppm (Portable Pixel Map) format images, so they may not display automatically on your computer, but you can open them in a standard image viewer if you'd like to take a look.

- maine1.ppm
- maine2.ppm
- maine3.ppm
- fallFoliage.ppm
- maineSnow.ppm

Objective 2 : Write a program to read and display an image

Create a simple program that will read image data from a file and display it in a window. Use the command line to specify the filename of the image to view.

First import both the *graphicsPlus* and *sys* packages. You now know multiple ways of importing a file in and creating aliases. I suggest you do so with the graphics library to make the code easier to write.

Then, create a main function in your show.py file. Give the main function one argument, argv, which will be the list of strings from the command line. One of these strings should be the filename of the image to open and view. The main function should do the following.

1. Check if the user provided enough command-line arguments
   Test if there are at least two strings in the main function parameter: the name of the python program and the filename of the image to open. If there are not at least two strings, print a usage message. You should know how to do this from previous assignments.

2. Use the Image function from the graphics library to read image data from a file and assign it to a variable

   Load image data from the file specified in the second string of the main function parameter into an Image. Assign to a variable (e.g. src_image) the result of calling the function Image() with a Zelle Point object (Point(0, 0)) as the first argument and the image's filename as the second argument. The filename will be the second element in the list argv that is the parameter of the main function. Assign the return value of the Image function to a variable. Make note that the Image() function and the Point functions are in the imported graphics library.

There are some class object stuff going on here that you will understand better after future modules. Use an assignment statement any time you need to store, retain, or assign information. In this case, we want to store the Image object created by the Image() function call. Therefore, put a variable name on the left side of an assignment and the Image() function call on the right.

3. Use the getWidth() and getHeight() methods of the Image object to get its size.

   Assign to width (or cols, your choice) the result of calling the image's getWidth method. Assign to height (or rows), the result of calling the image's getHeight method.

   Note that these methods are part of the image object that you saved in a variable.

4. Create a GraphWin window to display the image

   Using the width and height variables, create a GraphWin object with a title of your choice that is the same size as the Image. (Go look up this function in the provided file to learn more about how to use it properly.) Don't panic when you run and test. You won't be able to see the window until you "pause" the program.

5. Move the image to the center of the window

   Using the move method of the image, position the image in the center of the window. The center of the window is (width/2, height/2), and when creating the image object you used a point of (0, 0).

6. Draw the Image into the window

   Use the Image's draw method (just like any other graphics object) to draw the image into the window at its current anchor point. The draw method is part of an image object. Look it up to see how to use it properly.

7. Wait for user input

   Finally, call the getMouse method of the window. Use the name of the variable holding the window reference, then .getMouse() to call the method. This will wait for a mouse click in the window and then go on to the next instruction. If there is no next instruction, the program will terminate.

   Below the main function, put a conditional call to it. Pass the main function the list sys.argv.

```
 if __name__ == "__main__":
     main(sys.argv)
```

   This extra step will print the file from run if you import it into another application.

Once complete, try out your show function. On the terminal, make sure you are in the correct working directory and then run your program with an image filename as a command line argument. The following tries to display the image in flowers.ppm.

```
python3 show.py flowers.ppm
```

Objective 3 : Start a library of photo filters

Create a new file called filter.py. Do the usual stuff of putting your name and a date at the top. Then import graphics and sys. Save it in your project6 directory.

In filter.py, create a function named something like swapRedBlue that takes in one argument, which will be an Image object. The algorithm is written below as comments, properly indented. Read through them and make sure you understand the process. Then fill in the python code. After the Python code has been written, please remove these comments. They do not make the code any more readable, as they are just the English version of the code. Instead, put a comment in your own words above the function def statement.

The number of rows in the image is provided by src.getHeight(), if src is the Image object. The number of columns in the image is provided by src.getWidth(). If you wish, assign the height and width of the image to local variables like rows and cols.

```
def swapRedBlue( src ):
    # for each row i in the image
        # for each column j in the image
               # assign to (r, g, b) the pixel
               values at (j, i)
               # set the pixel indexed by (j, i)
               to the value (b, g, r)
```

You are going to want to research more functions from the imported code. Take a look at getPixel(...), setPixel(...), and color_rgb(..). The

TA will help you complete this section, but see if you can figure it out by yourself first.

It's best to loop over an image in row-major order (rows in the outer loop and columns in the inner loop) because that is how the image is stored in the computer's memory. Accessing memory in sequential order is always faster than jumping around.

Objective 4 : Wrapping up Test your photo filter

Test your filter function by going back to your show.py and calling it before you draw the image.

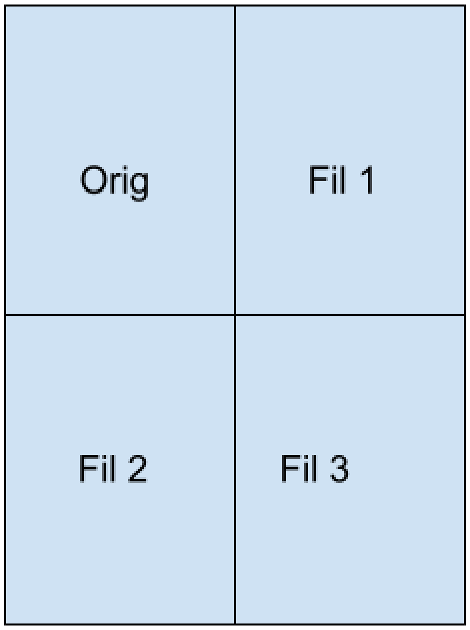
Review what you have created and ask your TA about anything you don’t understand.

Submit this code for your recitation ICE

3. **Lab Assignment Instructions:**

   Go to your project06 directory you created in recitation. You will be using your filter.py file from lab for the image filter effects. You'll also want to use your show.py file for inspiration. Create a warhol.py file that will be the main file for this assignment.

   Your ultimate goal is to create 4 versions of whatever image is sent to the warhol.py file in a Warhol art like fashion as a grid of 4. Do a quick web image search for Andy Warhol to see what I mean. Your warhol.py file should accept an image file, do the appropriate error checking and generate an image laid out similar to this:



You have all of the information you need to make this work, but you'll need to figure some things out to put it together. You'll need to make the window bigger, convert each image with a different filter, and place them accordingly. You might need to do some research or reference the GraphicsPlus library.

You may use the original and the filter created in class as two of the squares. The other two you will have to come up with on your own. Make these other 2 filters more complicated than swapping color channels. Some potential filters include ones that increase contrast, brighten or darken the image, generate a sepia effect, enhance the colors, or cartoonize the image. These filters should operate on a single pixel at a time. (If you want to implement a filter that uses more than one pixel, that is an extension.) Make the overall effect striking and unique. In other words, make sure it is obvious a filter was applied to the image.

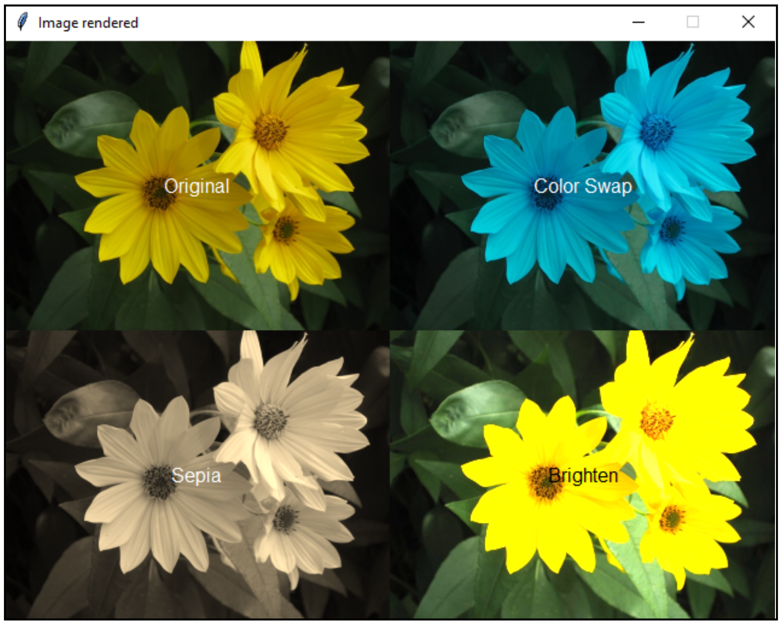
You'll need to research algorithms for accomplishing this, but they aren't too hard to find.

Lastly, add some text to each image. A short phrase, description, or your name will suffice. I just want to see you research one more graphics function that we did not cover in recitation.

You can create a Text object just like any other Zelle graphics object. The Text constructor takes a Point object as the first argument and the string to draw as the second argument. For example: gr.Text(gr.Point(x, y), 'some text'). Don't forget to draw the text into the window in order to see it. The text will be centered on the specified point, and you can control the size of the text using its setSize method and the color using its setFill method. If necessary, you can also modify the text using the setText method.

Don't try to do this all at once and test as you go. Set this up in stages. For example, get your grid working first, add one filter at a time, add text, work on extensions.

Here's what mine looks like. Make sure you do better. ;)



4. **Extension Suggestions:**

   Remember you can get up to 26/30 points for finishing the lab, but for 30 points you have to go above and beyond. You don't have to use any of the below extensions. They are just examples. Explore on your own and come up with something fun.

   Extension ideas:
   1. Create more than 4 squares.
   2. Implement more than 2 additional image transformations.
   3. Add additional creative elements like adding shapes to the image
   4. Enable the designer to pick the color or size of the text by adding information
   5. Enable it to handle more than one image in the final product
   6. Something else of your own design.

5. **Report:**

There was a lot of work in this assignment and I hope you created something you can be proud of. I just want a short report this week. This most important part is that you clearly point out the extensions that you did for the grader.

Reflection:

What was the easiest and hardest part of this assignment?
What did you learn?
What is your understanding/definition of a Pixel?
Why do we use nested for loops to process an image?

Images:

Images of you testing your final product.

Extension:

What extension(s) did you add to the assignment
Images of any extensions you successfully added

Grading Statement(optional):

What grade do you think you deserve and why?

6. **Submissions:**

You should be submitting these at the least, but you may submit more files if you separated out your assignment for organizational purposes:

- Report06.pdf
- warhol.py
- filter.py
- show.py (if needed)

Submit your project code on canvas as Lab6_"Your_name".zip

When you submit, double-check the following.

- ☐ Is your name and an appropriate header at the top of each Python file?
- ☐ Does every function have a comment or docstring specifying what it does?
- ☐ Does your report have all six sections completed?
- ☐ Is your report a pdf document?

7. **Rubric:**

	Possible	**Given**
Requested functions		
Images 1&2 (original and from project)	2	0
Image 3 (new filter pattern)	3	0
Image 4 (new filter pattern)	3	0
Neat grid pattern used	3	0
4 text phrases added	3	0
Misc		
Error checking	2	0
Lets you send in any file using CLI	2	0
Uses the filter file as requested	2	0
Code Quality	3	0
Report	3	0
Not included in total possible:		
Extensions (Not calculated without report)	4	0
Creative or went above and beyond	4	0
Code does not compile	-30	0
Late penalty	-6	0
Not implemented as requested	-30	0
TOTAL POINTS POSSIBLE out of 30	26	0