	EMBRY-RIDDLE AERONAUTICAL UNIVERSITY **Department of Electrical, Computer, and Software Engineering** **CS332 Organization of Programming Languages** (Spring 2019) Homework Exercise #3: Programming Language Control, I/O, and Types		
	PREPARED BY:	**LANGUAGE USED:**	**DUE DATE:**
	Gray, Andrea Rapp, Thomas	C and Python compiled with MobaXTerm GCC	22-FEB-2019

Note: This submission is tailored for Andrea Gray's code files and outputs specifically. The submission for Thomas Rapp is slightly altered to conform to his code and outputs.

Exercise 3 Requirements

1. Conceptually, the PSF, or Point Spread Function, appears rather simple to implement. Using the Guide to DSP [1], an acronym for Digital Signal Processing, the basic functionality of the program is laid out in just a few steps. The explanation offered for the creation of a PSF in the Guide to DSP is that the PSF is the convolution of an original pixel chosen at a particular weight with eight of the neighboring pixels subtracted from this value as a weighted sum. The weight, value k or $-\frac{k}{8}$, will first contrast the other eight pixels negatively with the original before a weighted sum is applied to enhance the contrast of the central pixel against its neighboring pixels. Upon reviewing the sample code, it seems as if the File I/O portion of the program will be the most challenging element as each file type has a very definitive formatting structure. While efficiency in any program is an essential concern, the efficiency of the implementation of the functions described in the Guide to DSP is not the primary focus due to the fact that, with this topic being new, most of the time spent coding and debugging will be for accurate compilation and execution.
2. For the implementation of the PSF Transform Thomas Rapp chose to use C programming language and Andrea Gray used Python programming language. Thomas chose C because the language is procedural, works well for high and low-level programming (to the extent needed for this assignment at least), and is well known by both himself and Andrea. Subsequently, the syntax for C is intuitive and is well versed for pointer arithmetic, which Thomas has a strong liking for due to its convenience, as well as compatibility between character arrays and strings which is helpful in indexing and modifying values cross-type. When the intentions to attempt to thread his program, Thomas instinctively chose C as it was the only programming language he knew how to do such a thing in. As for Andrea, she chose Python because she is most familiar with it after C and C++. When Andrea and Thomas decided to work together Thomas had already decided to work in C, and since Andrea needed to use a different language, she settled on Python. Although Python programming language was not her first choice, it is not a bad choice because Python, like PHP or Ruby, is a scripting language that is interpreted, object-oriented, and a high-level programming language with built in data-structures and dynamic typing and binding. This means that Python is not only emphasized in readability, but it also aids in programming productivity and ease of existing component connections. In comparison Python code is usually much more compact and concise than C which may be beneficial in execution, user debugging, and efficiency (although it is not the focus of this assignment).
   For the PSF transform for both Andrea and Thomas a Linux machine was used and the timing for the Python code is shown in Appendix A as Figure 1.2. The file used in the transformation is shown in Figure 1.0 with the transformation shown in Figure 1.1.
3. The routine for loading a PGM, Portable Graymap, into a two-dimensional array is included in this document submission as negative_pgm.py. The negative_pgm file for Python and the pgm file for

Python allows the user to change whether the code outputs a PSF image or a PGM image with or without the sharpening effect included. This is done through the user's alteration of the code by commenting and uncommenting sections. The test for the routine is shown below, from GIMP, in Figure 1.3 as the transformation from Figure 1.0. The binary comparisons of Figure 1.0 and Figure 1.3 are included in the file submission. The comparison and differences located by the binary editor, FrHEd, is also included with the submission files titled FrHEd_File_Compare.

4. Using the two-dimensional array created from step three above, a sharpen PSF is applied to each pixel of interest as an 8-bit unsigned type from a range 0-255. The new pixel obtained is stored in a transform array as a sum of products with each PSF multiplier applied to each of the eight nearest neighbors and K+1.0, the fourth element in the PSF, applied to the pixel selected. The sharpened image is shown in Appendix A: Figures as Figure 1.5 which is the Cactus-120Kpixel.pgm required to be used for this step (the original image is shown in Figure 1.4).
   Since Thomas used C as his programming language and is well versed in it (whereas Andrea has only been coding in Python for two months), he implemented a conditional expression on line 135 of the sharpen_PSF.c file also submitted with his document submission. The use of the conditional expression is for ensuring that the transformed image does not exceed the saturation range of 0-255 in his code and aided in its simplicity. This conditional expression was a slight challenge to Thomas even though he is well versed in C because the syntax is not commonly taught in introduction to C programming language courses and can be difficult to implement.
5. ****
6. The extra credit portion of this assignment is due to be turned in with the submission of Exercise #4.

Appendix A

Figures

The figures shown below are all from the execution of the C code. The figures submitted with Andrea's submission are the product of the Python code. They are submitted separately for conciseness with relation to the code created by each individual.

Figure 1.0: Base Image

Figure 1.1: PSF Transformation

Figure 1.2: PSF Time for Code – Python

Figure 1.3: PGM Routine Image

Figure 1.4: Original 120K Pixel

Figure 1.5: Sharpened PSF 120K Pixel

Figure 1.6: Original 12m Pixel

Appendix B

References:

[1] S. Smith, Ph.D., *The Scientist and Engineer's Guide to Digital Signal Processing*, Ch. 24. California Technical Publishing, 1997-2011. [E-Book] Available: http://www.dspguide.com/ch24/2.htm.