CSC 295 PROJECT 2

PPM P3 COLOR IMAGE BLENDING

THE ASSIGNMENT

This assignment will give you practice in file I/O and object-oriented programming. You will create a class definition for a Netpbm PPM P3 type image, read in two P3 color image data files, then create and output a blended color image.

NETPBM IMAGE FORMAT

The Netpbm image format options are summarized in the following table:

Netpbm Image Formats					
Type	Name	Magic Number		Extension	Colors
		ASCII	Binary (raw)		
PBM	Portable BitMap	P1	P4	.pbm	2 colors (0-1) white & black
PGM	Portable GrayMap	P2	P5	.pgm	256 colors (0-255) gray scale
PPM	Portable PixMap	P3	P6	.ppm	16,777,216 colors (0-255 for each RGB channel)

More information is provided at https://en.wikipedia.org/wiki/Netpbm#File_formats and in the slides provided on Moodle. Your program will focus on the P3 color image format in which the RGB color values are stored in the file as space-separated integers with a 255 max color value.

As shown in the slides, the format of a P3 image file begins something like:

```
P3
# Comment describing image or source
395 276
255
```

where the `P3` on the first line is the "magic number" that indicates an ASCII format color image, the second line includes a comment, the third line indicates the width and height of the image in pixels, and the fourth line indicates the max color code (255 in our case). After that, every 3 integers, valued between 0 and 255, provide the red, green, and blue color component of each pixel, respectively, starting at the top left corner of the image and moving across each row.

The sample files named `flag.ppm` and `soldiers.ppm` included with the files can be viewed as images using most image viewing software packages such as Photo viewer (Windows), Preview (Mac), or GIMP (GNU Image Manipulation Program). You might also try opening one of these in an editor so view the file format. After the initial 4 lines mentioned above, these image files contain 3 x 600 x 400 = 720,000 integers that describe the pixel color values.

CREATING A P3IMAGE CLASS

Create a class called `P3image` to hold the contents of a P3 image file. Include non-public attributes for each value included in the first 4 lines of the file, as well as a list to store the RGB pixel color values. Include property annotations for each of these attributes in lieu of accessor (getter) methods. Methods in the class should include:

1) An `__init__` method with 5 default parameters such that it can be passed all or some of the information stored in the first 4 lines of the file.

2)A `load_image` method that is passed the name of the image file in which the image is stored. It should open the file, read in the image data, store the data in the object and close the file. While the image comment is often the second line of the file, it could be any of the first 4 lines of the file. Your code should recognize the comment to prevent the image from being read in improperly or an exception from being thrown.

3) An `output_image` method that is passed the name of the file into which the image data should be dumped. This method should open the file, write the image data to the file, and close the file.

4) A `__str__ ()` method to override the class method to output the image comment, magic number, width, and height in the following format:

```
# blended image 60%/40%
Type: P3
Width: 600 Height: 400
```

You may include other methods as justifiable for your code organization and function.

MAIN PROGRAM REQUIREMENTS

Your `P3image` class should be contained in a separate source code file from your main program. Your main program should contain the code to instantiate the `P3image` objects and ask the user for the 2 image filenames used to create the blended image. It should also:

1) Contain a method external of the class called `blend_images` whose 2 arguments are the names of the image objects to be blended. It should return the image object holding the blended image. Instead of assuming a 50% / 50% blending ratio, prompt the user for the blending weight for the first image. Make sure that their input is a value between 0-100%. You can calculate the weight for the second image assuming a 100% total weight.

2) Inform the user of the name of the output file containing the blended image.

3) Output information on the blended image by calling the __str__ method.

OTHER PROGRAM REQUIREMENTS

Your code files should both be documented and should all handle exceptions. In the case of an I/O error due to a missing file or misspelled filename, allow the user to re-enter the filename until the file is found, so that your program can successfully execute and produce a blended image. When encountering a file with an inappropriate format, output a message to the user. In such a case, no blended image will be produced. You might find it helpful to import the Python `sys` module and terminate the program using the following statement when handling exceptions:

```
sys.exit("Execution terminated.")
```

The sample files included with the test files are of identical width and height. Your program should adjust for images of different dimensions in the way you deem best and produce an image.

You will want to test your program with a number of images. If you borrow an image that you find online or use one of your own, be aware that images larger than 1000 by 1000 pixels can take up a LOT of space. You might try using a lower resolution for cellphone photos.

If the image is in a JPEG, PNG, or HEIC format, it will need to be converted to a PPM P3 (ASCII) type. Simply changing the file extension DOES NOT change the file type. You might investigate your image viewing software for options on how to export it as a PPM ASCII format.

Zip your 2 Python source code files together so that they are contained in a single directory for grading. If you would like to include a final blended image from your own photos for **4 bonus points** you are invited do so.

SAMPLE INPUT/OUTPUT

Included in the test files directory are several images that you may use for testing. A few tests are described below.

Test Run 1:

```
Enter input filename for image 1: flag.ppm
Enter input filename for image 2: soldiers.ppm
Enter blending weight of first image as a percent: 60
```

Command Line Output:

```
Output file for blended image: imageBlend.ppm
# blended image 60.0%/40.0%
Type: P3
Width: 600 Height: 400
```

See image file output: `imageBlend.ppm`

Test Run 2:

```
Enter input filename for image 1: flag2.ppm
[Errno 2] No such file or directory: 'flag2.ppm'
Enter input filename again: flags.ppm
[Errno 2] No such file or directory: 'flags.ppm'
Enter input filename again: flag.ppm
Enter input filename for image 2: soldiers.ppm
```

Command Line Output:

```
Output file for blended image: imageBlend.ppm
# blended image 60.0%/40.0%
Type: P3
Width: 600 Height: 400
```

See image file output: `imageBlend.ppm`

Test Run 3:

```
Enter input filename for image 1: BWBeach.pgm
```

Command Line Output:

```
This file is not in PPM P3 format.
Execution terminated.
```

No image file output