CPS 111: Introduction to Computational Thinking
Homework 6 (20 points)

Handed out: October 24, 2018 (Wednesday)
Due: 11:59 PM November 1, 2018 (Thursday)
Late submissions accepted with penalty until 11:59 PM November 3 (Saturday)

If you run into trouble or have questions, arrange to meet with me or a tutor!

Before you submit:

- Please review the homework guidelines available on Canvas. Ensure that your code conforms to the style expectations set out in the homework guidelines.
    - Make sure your variable names are descriptive and follow standard capitalization conventions.
    - Put comments wherever necessary. Comments at the top of each module should include the file name, and a description of the module. Comments at the beginning of functions describe what the function does, what the parameters are, and what the return value is. Use comments elsewhere to help your reader follow the flow of your code.
    - *Program readability and elegance are as important as correctness.* After you've written your function, read and re-read it to eliminate any redundant lines of code, and to make sure variable and function names are intuitive and relevant.
- Read the assignment very carefully to ensure that you have followed all instructions and satisfied all requirements.
- Create a zip file named like this: *yourusernameHWX.zip* (with your username and replacing the *X* with the assignment number) that contains a directory named *yourusernameHWX*, which contains all of your .py files, and your cover sheet. **Make sure you have the entire directory in the zip archive, not just the files inside.**
- Ensure that your completed cover sheet is in the directory.
- Make sure to thoroughly test all of your programs before you submit your code.

Histogram equalization is an image processing method that attempts to relatively equally distribute pixel colors across the full range of possible colors. It adjusts the contrast in the image. This is particularly useful in images where there is low/poor contrast (e.g., satellite images). It is often used as a preprocessing step in other image processing operations such as object detection, image segmentation. The example below shows an image before and after equalization.

For this homework, you are going to implement histogram equalization for gray-scale images. The very broad idea is

1. Create a histogram of gray-scale values in the original image. Recall that gray-scale values will range from 0 to 255 and within a gray-scale image Red = Blue = Green.
2. Create a cumulative distribution from the histogram. It is a running sum of how often each gray-scale value occurs in the original image. You've done these first two steps before for the counting sort problem from lab.
3. Map gray-scale values from the original image to a new value based on the formula: newGray = $(cdf_{origGrayVal} - cdf_{min}) / (\text{area of the image} - cdf_{min}) * 255$. cdf_{min} is the minimum non-zero count from the cumulative distribution. Please note the final value for newGray needs to be an int.

But wait, don't write a single line of code yet. It's time to really start putting the computational problem solving approach to work. By Saturday 5 PM, you should submit (through Canvas) a text file containing any questions you might have regarding this problem and a plan for how to solve the problem. Your plan will be scored on a scale of 0 to 2 and feedback will be provided back to your

For your plan, you should fill in some of the details of the steps given above. Your plan should be detailed enough that another person should be able to translate it to mostly working code, but it should not contain code. Please note, that you should have your plan available any time you get help, especially from a professor or the tutors.

Write your code in a file called *equalize.py.* As far as your code goes:

1. You should have a `main` function that handles reading in the image as a command-line parameter, and displaying the original and equalized images in two separate windows.

2. You should have one function that returns an equalized image. You may assume the image is already in gray-scale. Do not modify the original image in any way.

3. You have been given *plotfreqs.py*. It is very similar to the *plotfreqs.py* you were given previously except I've made changes so that it only takes lists (instead of lists or dictionaries). You can still call the function with the second and/or third parameters set to `None`.

You can use it to plot the image histograms before and after equalization as an additional way to see if you are on the right track. You can also use it to plot the cumulative distribution. Below are the plots from the car example. The top two graphs are the histograms before and after equalization. You'll notice that gray-scale values are more evenly distributed in the second graph. The third graph is a plot of the cumulative distribution of gray-scale values from the original image.