Assignment 1

Name: Aman Bhargava

Part I Answer

Question: If the two files you compared above are the same, does it prove that your code is correct? Explain.

Answer: Strong similarity give a good hint that the code is *probably* correct, but it by no means proves it. I may, for instance, have made a mistake in the way was assigning values during my computation resulting in the copying of a an incorrect result to both files. As well, the files are saved differently (CSV vs. npy) -- depending on what 'comparison' and 'the same' mean, I may arrive at an incorrect conclusion. If I were to examine the contents of the files in an intelligent fashion and take into account the fact that the floating point arithmetic may result in slightly different results, then I could get a decent sense of the correctness. At no point, however, would I be able to 'prove' the correctness of my code through any observation of the output.

Part II Answer

Question: What is the output in the terminal from `part2_test.ipynb`?

Answer:

```
Input:  [ 10   5  -5 -10]
Output: [-1.5 -2.8  1.6 12.8]
```

Part III Answers (Qualitative Questions)

1. **How does `img_add.png` differ from the original image? What would happen if we had subtracted 0.25 from the original image instead of adding?**

`img_add.png` differs in that it is brighter and that it has *less information* contained in it due to the clipping. If we had subtracted 0.25 from the original image, it would have been darker, and there would have also been information loss (potentially a different amount of information loss depending on the distribution of the pixel values).

2. **Describe your programming experience in a few paragraphs.**

   Other than the conventional year 1-2 Engineering Science courses, I was a software engineer at a technology consulting company where I personally handled several of their software contracts (generally using Javascript web frameworks and various back-end-as-a-service systems) and robotics contracts. I also started my own medical analytics company where we designed and developed a front-end data collection and data visualization system using similar Javascript frameworks to before. The back-end was primarily written by Adam Carnaffan (CareTrack.io's head of development) in Flask.

   More recently, I was a researcher at MannLab where I developed expansive real-time biometric signal processing systems using Scipy, Numpy, and Matplotlib (also Kivy for a front-end UI) to deduce biometrics via PPG, radar, rPPG (blood flow detection via conventional RGB camera data), and accelerometer. The paper I helped write about these systems has been accepted as an elevated paper presentation at IEEE Sensors 2020. I also created a real-time brain scan (EEG) processing system that utilizes deep reinforcement learning to guide a user's meditation via real-time musical modulation. This project utilized PyTorch, Numpy, Scipy, asynchronous Python functionality, Python UDP servers, microprocessor programming (ESP32), and a C library called portaudio for interfacing at a low-level with audio drivers to create the user's musical experience. The paper I wrote for this has been accepted into IEEE Sensors 2020.

   My latest programming experience was a novel signal processing library I wrote to classify brainwave data. I created an improved efficient, parallelized, version of the Adaptive Chirplet Transform from the ground up in Python

using Numpy for matrix computation and Scipy for optimization. This was in order to create a machine learning classification pipeline for the P300 brainwaves that could out perform the conventional classification pipelines (I used Scikit-Learn for the ML classifiers). The system I created out performed the conventional methodology and the paper is in its final stages of review before submission to IECBES2020.

3. **Describe your experience with Assignment 1.**

   Assignment 1 was well-written and easy to understand. There were only a couple places where I was unclear:

   1. Is it OK to use alternative methods to achieve the same outcomes? For instance, I utilized Numpy array indexing and the `numpy.copy` function to extract each of the R/G/B channels instead of initializing an array of zeros first and copying the values over.
   2. For the grayscale image, I specified the colormap as `gray` so that it was actually gray when I exported (not yellow and blue as in the default `viridis` colormap). This was not, however, specified (as far as I saw), so I had to make this assumption.