Applications in Scientific Computing
Assignment 2: Image processing

530.390.13

Due: Friday 8 January 2016

1. Convert the binary number 1010 0011 by interpreting it as both 8-bit signed and unsigned integers.

   - `int8(1010 0011)` = -128 + 32 + 2 + 1 = -93
   - `uint8(1010 0011)` = 128 + 32 + 2 + 1 = 163

2. Working from the code we wrote in class (i.e., use no built-in functions), write a routine for generating the "lightness" grayscale, defined by:

$$R_{i,j} = G_{i,j} = B_{i,j} = \frac{1}{2}\left[\max\left(R,G,B\right)_{i,j} + \min\left(R,G,B\right)_{i,j}\right].$$

   Note that you will also need to modify the maximum and minimum functions from class. Commit this new code to your personal GitHub repository for the course; simply turn in your username and the path to the new code in the repository.

   - See the companion Python module file from class 2, entitled `assignment-2.py`.

3. Imagine you must search an array for some data. You have two options: you may sort the data so that you can use "smart" searching algorithms (like our binary search), or you may simply search the data as-is (like our unsorted search). Can sorting and searching possibly be faster than searching without sorting? What are the benefits and drawbacks of option?

   - Sorting before searching: Searching is $O\left(\log N\right)$, but requires an $O\left(N \log N\right)$ sort (if using merge sort, e.g.)
   - Unsorted searching: Searching is $O\left(N\right)$
   - If searching is only required one time, it is best to search without sorting because the sorting alone will take longer than $O\left(N\right)$. However, if searching must be performed more than one time, it will become more efficient to sort first, though the exact point at which $O\left(N\right) > O\left(N \log N\right) + nO\left(\log N\right)$ depends not only on the number of searches n, but also on the sorting and searching implementations.