CMSC 427 Computer Graphics
Programming Assignment 2: Image Processing
Due Date: 11:59 PM, March 2, 2017
Project Submission:

1) Delete all intermediate files (run the command `make distclean`) and the Makefile created during the compilation process. Also delete the executable file.
2) Place all files for the project in a folder and ZIP up the folder. You will submit your project via the submit server. To submit a zip file, login to the submit server webpage and look for the link to make a *web submission*.

Project Description

In this project, you must choose operations from the following list to implement. These operations have been divided into groups. Within each group, you may choose what operations to implement. At a minimum, you must implement the required number of operations per group. **You may not use the image processing operations implemented in Qt's beyond calls to pixel (http://doc.qt.io/qt-5/qimage.html#pixel-1), setPixel (http://doc.qt.io/qt-5/qimage.html#setPixel), QColor (http://doc.qt.io/qt-5/qcolor.html), and the QImage constructors.**

- Luminance Operations **(required: implement 2 of your choice)**
  - Brightness: Change the brightness of the image by a real-valued alpha factor. See Graphica Obscura here < http://www.graficaobscura.com/interp/index.html >.
    - The alpha factor can be any value in the range [0.0, 2.0].
  - Contrast: Change the contrast of an image by a real-valued alpha factor. See Graphica Obscura here < http://www.graficaobscura.com/interp/index.html >.
    - The alpha factor can be any value in the range [-1.0, 2.0].
  - Gamma: Apply a gamma correction to the image using a specified gamma. See here for reference < https://en.wikipedia.org/wiki/Gamma_correction >
    - The gamma can be any value in the range (0,5).
- Color Operations **(required: implement 1 of your choice)**
  - Black & White: Convert to gray levels by replacing each pixel with its luminance.
  - Saturation: Change the saturation of an image using a specified saturation factor. See Graphica Obscura here < http://www.graficaobscura.com/interp/index.html >
    - The saturation factor can be any value in the range [-1.0, 2.5].
- Filtering Operations **(required: implement 3 of your choice)**
  - Bilateral Filter: Smooth while preserving sharp edges in the original image. See here http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MANDUCHI1/Bilateral_Filtering.html and here < http://people.csail.mit.edu/sparis/bf_course >.
  - Gaussian Blur: Blur an image by convolving it with a low-pass filter (a.k.a Gaussian blur) given a real-valued sigma.
    - The sigma can be any value in the range [0.33, 5.0]
    - The radius for this operation will be defined as $r = 3*\text{sigma}$.
  - Channel Extract: Leave the specified channel intact and set the other 2 channels to zero.
    - You should modify the RGB channels, but do not change the alpha value.
  - Motion Blur: Apply a left-right linear blur to the image given an integer-valued magnitude.
    - The input parameter magnitude represents the radius of the filter that is used. More magnitude (which will create a larger filter) represents more blurring.
    - The magnitude can be any real value in the range [1, 100].
  - Median Filter: Remove speckle noise using a median filter of a given width. See here

< https://en.wikipedia.org/wiki/Median_filter >.
    - Sharpen: Apply a linear sharpening filter to the image.
        - See class slides for filter example.
- Translation Operations **(required: implement both, see note below regarding sampling methods)**
    - Rotate: Rotate an image around the center by a given angle.
        - The angle will be specified as a real-value in the range [0, 360].
    - Scale: Scale an image up or down in the x and y direction by a real valued factor.
        - The factor can be any value in the range of [0.05, 20]. This value range should produce images as small as 1/20 of the original size to images that are 20 times larger than original size).
- Transformation **(required: implement 1 of your choice)**
    - Crop: Remove the outer part of an image. The crop window is specified by input parameters where
        - x: the x coordinate of the top-left corner of the crop window
        - y: the y coordinate of the top-left corner of the crop window
        - width: the width of the crop window
        - height: the height of the crop window
    - Composite: Compose one image with a second image using the over operator, with a third image as a matte (alpha).

For more details on how input parameters are entered for each operation run the program with the -help flag.

Note: You must implement 3 sampling methods (point, linear, and Gaussian sampling), which will be controlled by the -sampling option prior to applying -scale, -rotate, or –fun to an image. For Gaussian sampling, choose sensible Gaussian filter dimensions. For -scale, this requires adapting the extent of the Gaussian. For -rotation and -fun, you may use a fixed size of your choosing.

Extra credit – 5% extra credit
Implement one of the following:

- Fun: Warp an image using a non-linear mapping of your choice (examples are fisheye, sine, bulge, swirl).
- Nonphotorealism: Implement any non-trivial painterly filter. For inspiration, take a look at the effects available in programs like xv, *PhotoShop*, *Image Composer*, and *GIMP* (e.g., impressionist, charcoal, stained glass, etc.). The points awarded for this feature will depend on the creativity and difficulty of the filter. At most one such filter will receive points.

Code Requirements
You may not use any functions implemented in the QImage class other than those necessary to access/modify pixel values. You will be graded on the completeness of your implementation of the algorithms above. If unsure about a QImage function, ask about it on Piazza.

You may add helper methods to the code, but you are prohibited from changing the command line options. You program should be able to run using the options defined in the –help menu.

Your program must be executable from the command line using qmake and make. It should be capable of reading and writing images in the JPG format. To execute your code from the command line, please read the comments inside the README.txt file.

Project Grading
The TA will use an undisclosed set of images and command line options to test your implementation so we encourage you to find more images online in JPG format to test your program. We have provided you with three JPG images for testing purposes.

Your program will be graded based upon successful calls to it from the command line and a correct implementation. Your code should be well commented (especially in the header files). Note that it will also be judged subjectively based on the simplicity and clarity of implementation. An implementation that is easy to understand, but has few minor bugs will be scored higher than a messy implementation with the same number of minor bugs.

Documentation Grading

Please include documentation in the form of a Markdown (https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet) document with your assignment. It should be a text file in the folder ./cmsc427 called DOCUMENTATION.md. You will be graded on the completeness, clarity, and conciseness of your documentation. Your documentation should links to inline images showing the before and after of your image processing operations. It should also include an explanation of the algorithm you implemented. The more complete and clear your explanation is in this documentation the easier it will be for us to assign partial credit. It is in your interest to be as concise and clear as possible.