Assignment 1 – Image Processing

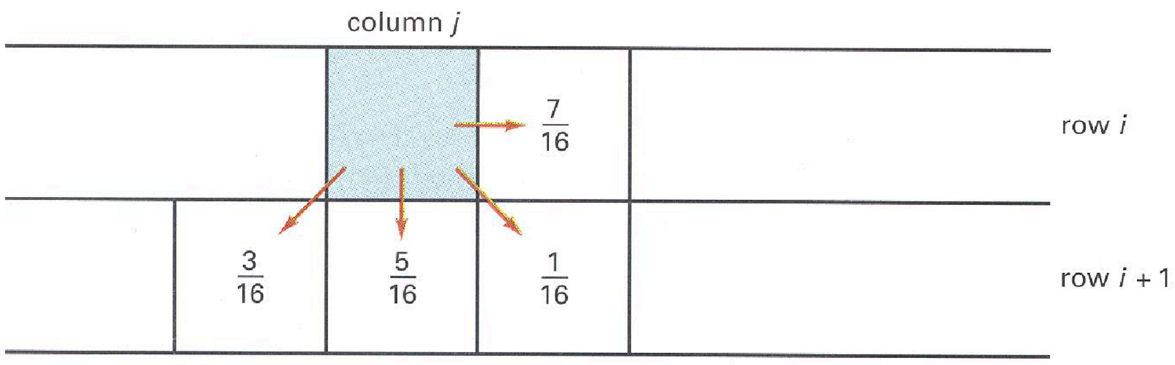
Overview:

In this assignment you will implement image filters using C++ or Python (Depending on your choice). The input is a RGB **.png** image of the famous Lenna located in the assignment zip file.

Main task:

Read the RGB image of Lenna and create 4 png images as follows:

1. **Grayscale.png** - square shows the image in **Grayscale**
   (each pixel has a value between 0 (black) to 255 (white)).
2. **Canny.png** - square shows the image **Edges** using **Canny Edge Detection**
   (each pixel has a value of 0 (black) or 255(white)).
   Guidance: Implement all the following stages:
   - Noise reduction using **Gaussian Filter**.
   - **Gradient Calculation** along the horizontal and vertical axis.
   - **Non-Maximum Suppression** of false edges.
   - **Double thresholding** and Edge Tracking using **Hysteresis**.
3. **Haftone.png** - square shows the **Halftone** pattern
   (each pixel becomes 4 pixels in the new black and white picture).
   Guidance: Start with creating an array or matrix with a size of 512x512, and remember to run each time on 2 rows at the same time.
4. **FloyedSteinberg.png** - square implements **Floyd-Steinberg** Algorithm, which changes the image from 256 intensity grayscale values to 16 intensity grayscale values.
   Guidance: Use the following **Error Diffusion Dither** formula (if you have missing edges, split the error equally between the available neighbour pixels)



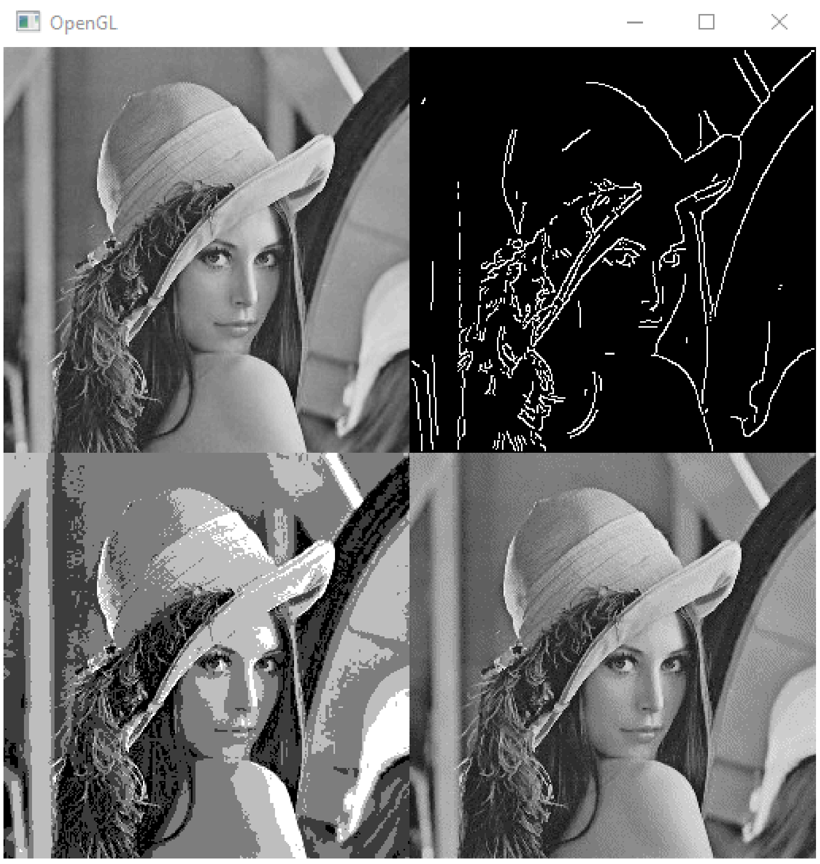
Debug:

Create 4 text files for all the 4 images. Each file will contain just numbers separated by commas (without spaces).
The numbers will represent the pixel values (0-1 for black and white picture, and 0-15 for grayscale picture).
The names of the files will be: **Grayscale.txt**, **Canny.txt**, **Halftone.txt**, and **FloyedSteinberg.txt**.
These files will be generated in the project/assignment folder.

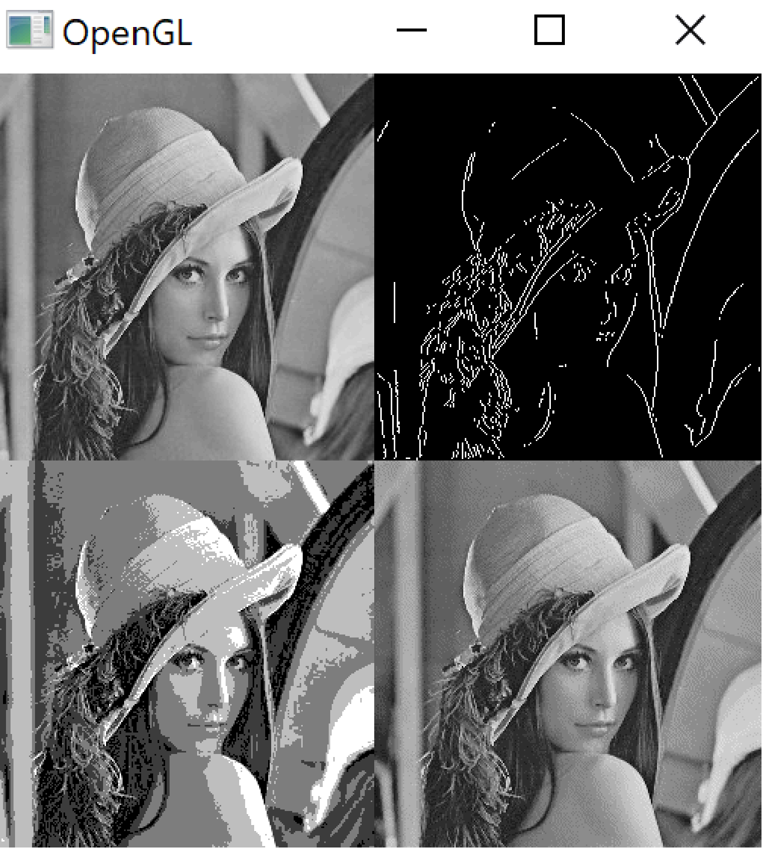
Notices about the example output in the next page:

1. The Canny Edge Detection might look different from the example outputs in this assignment depending on the Data Structures and Filters you will be using.
2. The Halftone images in the example outputs are resized to a resolution of 256x256 from 512x512, to show all the 4 images on the same window. You may check if your implementation is correct if you resize your Halftone image to the same resolution and get a similar result.
3. The order of the outputs from top-left to bottom-right:
   a. Grayscale
   b. Canny Edge Detection
   c. Halftone
   d. Floyd-Steinberg

Example of expected output:



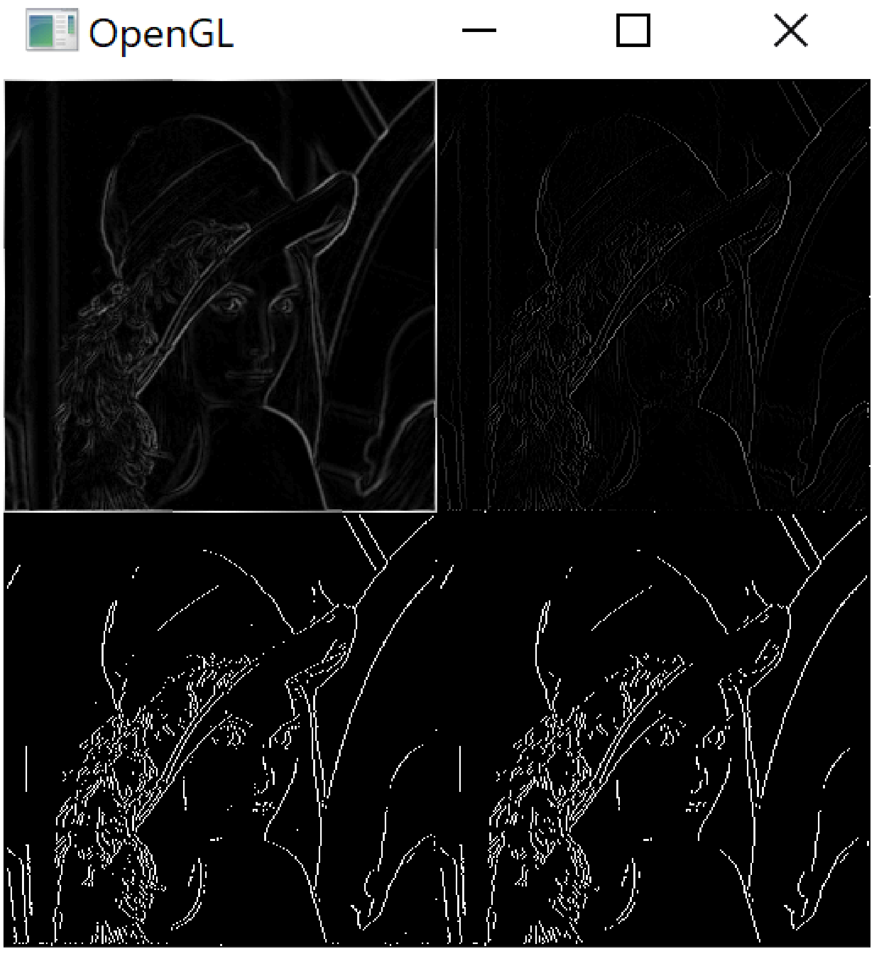
Another example of expected output:



Stages of Canny Edge Detection:

For debugging purpose, the images show the expected output after each stage in **Canny Edge Detection**, from top-left to bottom-right:

1. Gradient Calculation
2. Non-Max Suppression
3. Double Thresholding
4. Hysteresis



General tips (for C++):

1. When you read the image with the function "stbi_load", you get an unsigned array (unsigned char *) where each pixel is composed of 4 values (RGBA).
   To convert the values from 4 values (RGBA) to 1 value (Grayscale) you can either:
   a. Take the average of the 3 RGB color channels
   b. Use a smarter way to convert the image to Grayscale.
2. Before plotting a pixel, make sure that its color does not exceed the range of 0-255 for every color channel. Use a function "clip" to handle that, otherwise your pixel colors will be interpreted as the invert colors, for example: 260 will get interpreted as 5, and -5 will be interpreted as 250.
3. In Halftone you need to create an image that is bigger than the original image by 4, so use the function "malloc" to allocate the required new memory.

<u>**Assignment Score:**</u>

- **Sanity** - Grayscale: **5 points**
- Canny Edge Detection: **35 points** (5 from 35 points are: Impress me)
- Halftone: **20 points**
- Floyed Steinberg: **20 points**
- Performance: **20 points**

<u>**Submission:**</u>

Submit zip file with the following files:

1. Link to your GitHub repository.
2. **(Optional)** Text, Doc or PDF file with short explanation about the changes you did in the engine (files you change and functions you modify).
   This file will help you and us to understand what changes you have made in the engine to complete the assignment 😊

The zip file name must include your ID numbers as follows: **ID1_ID2.zip**.