CS4551 Multimedia Software Systems (Spring 2019)

Homework1 (3%) – Getting Familiar with Template Code

- Due: Electronic submission via CSNS by **Wednesday, 02/13/2019**.
- What to turn in:
  - Submit ***source code*** with necessary files for "compile and run".
  - Do NOT submit data files.
  - You **MUST** provide a ***readme.txt*** file containing all information to help with the grading process.
- If your program produces any compile errors, you will receive 0 automatically no matter how close your program is to the solution.

- Programming requirements:
  - **You are not allowed to use any Java built-in image class methods, library, or tools to complete this homework.**
  - Do not create one mega-size main class.
  - **Do not change any given methods of `MImage` class nor create a new class that duplicates `MImage` class. Treat `MImage` as a part of imported library.**
  - Test your program with all test data.
  - If you do not meet any of the requirements above, you will receive a significant reduction.

0. What your program should do

Name your main application ***CS4551_[YourLastName].java*** (eg. CS4551_Doe.java) and expand the given template program to perform the following tasks.

Receive the input file as command line arguments.

<eg> On Command Prompt

```
java CS4551_Doe Ducky.ppm
```

Read a 24bit input PPM image and display the following main menu to the user and receive the user's input.

```
Main Menu-----------------------------------
1. Conversion to Gray-scale Image (24bits->8bits)
2. Conversion to 8bit Indexed Color Image using Uniform Color
   Quantization (24bits->8bits)
3. Quit

Please enter the task number [1-3]:
```

After performing a selected task, go back to display the menu again.

1. Task 1 - Conversion a 24-bit Color to a 8-bit Gray-scale (20 pts)

Convert the input (24-bit color pixels) to gray-scale pixels. Use the following equation to compute the gray-scale value from R(Red), G(Green), and B(Blue) of each pixel.

```
Gray = round(0.299 * R + 0.587 * G  + 0.114 * B)
```

After the computation, make sure that `Gray` is an integer ranging between 0-255 inclusive. Truncate if it is not in the range. Save gray values into a PPM file. Read the supplementary reading material to see how to save 8bit gray-scale valued pixels in PPM format.

2. Task 2 - Uniform Color Quantization (80 pts)

Quantize the input (24bit color pixels) to 8bit colors using the **UCQ (Uniform Color Quantization)** method.

Step1 (10 pts): Generate and display the 8-bit color Look Up Table (LUT)
Step2 (40 pts): Convert the input (24-bit pixels) to 8-bit indexed values and save index values in PPM format.
Step3 (30 pts): Save the quantized RGB pixels in PPM format.

See implementation detail about each step below.

Step1: Generate the LUT by the UCQ and display the table to the console as shown below.

```
LUT by UCQ
Index       R     G     B
_____________________________
0           16    16    32
1           16    16    96
2           16    16    160
...
5           16    48    96
…
255         .     .     .
```

Step2: For each 24-bit pixel of the input, calculate an 8-bit index value. Save the 8-bit index values into `[InputFileName]-index.ppm` file. Read the supplementary reading material to see how to save 8bit gray-scale valued pixels in PPM format.

Step3: For each 8-bit index value of the input, retrieve (or get) a RGB value from the LUT. Save retrieved RGB values into a PPM file named `[InputFileName]-QT8.ppm`.

For all the tasks, do not overwrite the original input in order to save your output.
See HW1 Sample Results