Hack 6.0

Computer Science I

Department of Computer Science & Engineering
University of Nebraska–Lincoln

Introduction

Hack session activities are small weekly programming assignments intended to get you started on full programming assignments. Collaboration is allowed and, in fact, *highly encouraged*. You may start on the activity before your hack session, but during the hack session you must either be actively working on this activity or *helping others* work on the activity. You are graded using the same rubric as assignments so documentation, style, design and correctness are all important. This activity is **due at 23:59:59 on the Friday** in the week in which it is assigned according to the CSE system clock.

Problem Statement

In this hack you'll get some more practice writing functions that utilize pass-by-reference (pointers), error handling and enumerated types. There are several different ways to model colors including RGB and CMYK. RGB is generally used in displays and models a color with three values in the range $[0, 255]$ corresponding to the red, green and blue "contribution" to the color. For example, the triple $(255, 255, 0)$ corresponds to a full red and green (additive) value which results in yellow. CMYK or Cyan-Magenta-Yellow-Black is a model used in printing where four colors of ink are combined to make various colors. In this system, the four values are on the scale $[0, 1]$. Write several functions to convert between these models as well as converting an RGB color to a *grayscale* value.

1. Write a function to convert from an RGB color model to CMYK. To convert to CMYK, you first need to scale each integer value to the range $[0, 1]$ by simply computing

$$r' = \frac{r}{255}, \quad g' = \frac{g}{255}, \quad b' = \frac{b}{255}$$

and then using the following formulas:

$$\begin{aligned} k &= 1 - \max\{r', g', b'\} \\ c &= \frac{(1 - r' - k)}{(1 - k)} \\ m &= \frac{(1 - g' - k)}{(1 - k)} \\ y &= \frac{(1 - b' - k)}{(1 - k)} \end{aligned}$$

Your function should have the following signature:

```
int rgbToCMYK(int r, int g, int b, double *c, double *m, double *y, double *k)
```

Identify any and all error conditions and use the return value to indicate an error code (0 for no error, non-zero value(s) for error conditions).

2. Write a function to convert from CMYK to RGB using the following formulas.

$$\begin{aligned} r &= 255 \cdot (1 - c) \cdot (1 - k) \\ g &= 255 \cdot (1 - m) \cdot (1 - k) \\ b &= 255 \cdot (1 - y) \cdot (1 - k) \end{aligned}$$

Your function should have the following signature:

```
int cmykToRGB(double c, double m, double y, double k, int *r, int *g, int *b)
```

Identify any and all error conditions and use the return value to indicate an error code (0 for no error, non-zero value(s) for error conditions).

3. Write a function that takes an RGB color value and transforms it to a grayscale version, "removing" the color values. An RGB color value is grayscale if all three components have the same value. To transform a color value to grayscale, there there are several possible techniques. The average method simply sets all three values to the average:

$$\frac{r + g + b}{3}$$

The lightness method averages the most prominent and least prominent colors:

$$\frac{\max\{r, g, b\} + \min\{r, g, b\}}{2}$$

The luminosity technique uses a weighted average to account for a human perceptual preference toward green, setting all three values to:

$$0.21r + 0.72g + 0.07b$$

In all three cases, the integer values should be *rounded* rather than truncated.

Your function should have the following signature:

```
int toGrayScale(int *r, int *g, int *b, Mode m)
```

To specify which of the three *modes* is to be used, define and use the following enumerated type in your header file:

```
typedef enum {
  AVERAGE,
  LIGHTNESS,
  LUMINOSITY
} Mode;
```

Identify any and all error conditions and use the return value to indicate an error code (0 for no error, non-zero value(s) for error conditions).

Instructions

Place all of your function prototypes into a file named `utils.h` and and their definitions in a file named `utils.c`. In addition, create a main test driver program that demonstrates at least 3 test calls to each of your functions to verify that their output matches your expected output. You should *not* prompt for input; rather your test cases should be hardcoded in your test driver's main function. Place your main function into a file named `utilsTester.c`. Hand in all three of your source files using webhandin.

- You are encouraged to collaborate any number of students before, during, and after your scheduled hack session.
- Design at least 3 test cases for each function *before* you begin designing or implementing your program. Test cases are input-output pairs that are known to be correct using means other than your program.
- You may (in fact are encouraged) to define any additional "helper" functions that may help you.
- Include the name(s) of everyone who worked together on this activity in your source file's header.
- Place your prototypes and documentation in a header file named `colorUtils.h` and your source in a file named `colorUtils.c`.
- In addition, implement all of your test cases using cmocka (`https://cmocka.org/`), a unit testing framework for C. An starter file, `utilsTester.c` has been provided which gives several examples of tests for the `rgbToCMYK` function. You should add your own as well as use the rest of the code as a guide for implementing tests for the other two functions. The starter file should be sufficient, but the full

documentation can be found here: https://api.cmocka.org/. A `makefile` has also been provided to help you easily compile your files. Note that cmocka is already installed on the CSE server. If you use your own machine, you will have to install and troubleshoot cmocka yourself.

- Turn in all of your files via webhandin, making sure that it runs and executes correctly in the webgrader. Each individual student will need to hand in their own copy and will receive their own individual grade.