CSCE 240 – Programming Assignment Two

Due: 11:59pm on Thursday, September 12th

Course Purpose – Demonstrate the ability to write reusable functions and appropriately use value parameters, reference parameters, and default arguments by implementing the functions described below.

Programming Purpose – Create a collection of color representation functions described below. The functions include rough conversions (using the formulas provided below) between RGB and CMYK.

The Red, Green, and Blue (RGB) values must be integers between 0 and 255, inclusive.

The Cyan, Magenta, Yellow, and blacK (CMYK) values will be decimal numbers between 0 and 1, rounded to the hundredths place, that represent the percent of cyan, magenta, yellow, and black.

RGBtoCMYK – This function takes seven arguments. The first three arguments are integers holding the Red, Green, Blue representation of a color, respectively. The next four arguments are double variables for an equivalent Cyan, Magenta, Yellow, blacK representation of the color.

If the RGB values are valid (each between 0 and 255), the function will set the Cyan, Magenta, Yellow, and blacK arguments using the formulas provided below, and the function will return true.

If the RGB values are invalid, the function will leave the Cyan, Magenta, Yellow, and blacK arguments unchanged, and the function will return false.

The algorithm below approximates C, M, Y, and K using RGB values.

```
K = 1 - max( R/255, G/255, B/255 )

if K is 1
     C = M = Y = 0
else
     C = ( 1 - R/255 - K ) / ( 1 - K )
     M = ( 1 - G/255 - K ) / ( 1 - K )
     Y = ( 1 - B/255 - K ) / ( 1 - K )
```

C, M, Y, and K should be rounded to the hundredths place.

For example, RGBtoCMYK(204, 186, 37, c, m, y, k) should set c to 0, m to 0.09, y to 0.82, k to 0.2, and should return true.

RGBtoCMYK(115, 260, 184, c, m, y, k) should leave the variables c, m, y, and k unchanged, and should return false.

For more examples, see the sample tests provided in testRBGtoCMYK.cc

CMYKtoRGB - This function takes seven arguments. The first four arguments are doubles holding the Cyan, Magenta, Yellow, blacK representation of a color, respectively. The next three arguments are integer variables for an equivalent Red, Green, Blue representation of the color.

If the CMYK values are valid (each between 0 and 1), the function will set the Red, Green, Blue arguments using the formulas provided below, and the function will return true.

If the CMYK values are invalid, the function will leave the Red, Green, Blue arguments unchanged, and the function will return false.

The formulas below approximates R, G, and B using CMYK values.

```
R = 255 * (1 - C) * (1 - K)
G = 255 * (1 - M) * (1 - K)
B = 255 * (1 - Y) * (1 - K)
```

R, G, and B should be rounded to the closest integer.

For example, CMYKtoRGB(0, 0.09, 0.82, 0.2, r, g, b) should set r to 204, g to 186, b to 37, and should return true.

CMYKtoRGB(0.32, 0.74, 1.31, 0.04, r, g, b) should leave the variables r, g, and b unchanged, and should return false.

For more examples, see the sample tests provided in testCMYKtoRGB.cc

EquivalentColors - This function takes eight arguments. The first three arguments are integers holding the Red, Green, Blue representation of a color. The next four arguments are doubles holding the Cyan, Magenta, Yellow, blacK representation of a color. The eighth argument is an integer, with a default value of 0, which holds the allowable R, G, and B margin of error when comparing the two colors.

The function should compute the RGB equivalent to the CMYK color using the formulas provided for the CMYKtoRGB function above. If the values of the Red, Green, and Blue arguments sent to the function are within the margin of error of the converted RGB color, the function should return true. Otherwise, the function should return false.

For example,

```
      EquivalentColors(204, 186, 37, 0, 0.09, 0.82, 0.2)
```

should return true since the CMYK color C=0, M=0.09, Y=0.82, K=0.2 will convert to the RGB color R=204, G=186, B=37.

EquivalentColors(205, 185, 37, 0, 0.09, 0.82, 0.2) should return false since the CMYK color C=0, M=0.09, Y=0.82, K=0.2 to the RGB color R=204, G=186, B=37, and the first three arguments do not match these R, G, B values.

EquivalentColors(205, 185, 37, 0, 0.09, 0.82, 0.2, 1) should return true since the CMYK color C=0, M=0.09, Y=0.82, K=0.2 to the RGB color R=204, G=186, B=37, and the red, green, and blue arguments are each within 1 (the margin of error sent as the eighth argument) of those values.

OutputAsHex - This overloaded function will take either three integer arguments holding the R, G, and B representation of a color OR four double arguments holding the C, M, Y, and K representation of a color.

If the argument values are valid, the function should output the hex representation of the color to the standard output device (using cout) and return true.

The hex representation of a color is 7 characters in the format

#rrggbb

where rr is the two-digit hexadecimal representation of R, gg is the two-digit hexadecimal representation of G, and bb is the two-digit hexadecimal representation of B.

For example, OutputAsHex(5, 15, 234) should output

#050FEA

to the standard output device, and return true.

OutputAsHex(0, 0.9, 0.82, 0.2) should output

#CCBA25

to the standard output device, and return true.

For a refresher on converting between positional numeration systems like decimal, hexadecimal, binary, etc, see the attached NumerationSystems PDF.

If the function is sent invalid RGB arguments, it should output

“Invalid. RGB values must be between 0 and 255.”

to the standard output device, and the function should return false. For example, OutputAsHex(28, 17, -10) should return false and display the invalid output message.

If the function is sent invalid CMYK arguments, it should output

“Invalid. CMYK values must be between 0 and 1.”

to the standard output device, and the function should return false. For example, OutputAsHex(0.31, 0.28, 1.46, 0.91) should return false and display the invalid output message.

Note: the final output for this function must be an endl.

Additional Specifications

- All function prototypes must be contained in a file named *program2functions.h*
- All function implementations must be written in a file named *program2functions.cc*
- You may write more functions in addition to the required functions listed above. All user-defined functions used by the functions required for this project must be written in *program2functions.cc*c
- The only header files that can be included in your code are iostream, cmath, and your program2functions.h. Files that include other headers will not be eligible for correctness points.
- You will submit your *program2functions.h* and *program2functions.cc* files to the assignment in Blackboard.
- cpplint error messages regarding the use of non-const reference parameters are expected and are to be ignored. Cpplint error messages regarding including the path (for includes and in the header guards) are also expected and should be ignored. All other cpplint errors should be addressed (i.e. cpplint should find no other errors in your files).
- Programs must compile and run on a computer of the instructor's choosing in the Linux lab (see your course syllabus for additional details).
- Be sure to review the program expectations section of the course syllabus.

Initial Testing

Initial tests for the functions are attached to the assignment in Blackboard. A *makefile* has been included to run your functions with the sample tests. In order to use the *makefile,* ensure that your *program2functions.h* and *program2functions.cc files* and all of the files attached to the assignment are in the same directory. Your program will be graded using this same method with modified tests.

The commands to run the sample tests are given below:

```
make testRGBtoCMYK
make testCMYKtoRGB
make testEquivalentColors
make testOutputAsHex1
make testOutputAsHex2
```

You are encouraged to create additional, more rigorous tests.

Grade Breakdown

Style: 1 point
Documentation: 1 point
Clean compile of *program2functions.cc*: 1 point
RGBtoCMYK passes instructor's tests: 1.5 points
CMYKtoRGB passes instructor's tests: 1.5 points
EquivalentColors passes instructor's tests: 1 point
OutputAsHex with three arguments passes instructor's tests: 1.5 points
OutputAsHex with four arguments passes instructor's tests: 1.5 points

The penalty for late assignment submissions is 10% per day up to three days after the assignment due date. No assignment submissions will be accepted more that 3 days after the due date.