Programming Assignment 2: C Kernels (100 points)

The goal of this assignment is to become familiar with C kernels. A kernel is a function or library you call from a higher level language, such Python or Java. Your task is to write a C kernel,an equivalent Python function, and measure the performance differences between them.

The kernel will convert an image in RGB format to YCrCb format. An image stored in RGB format stores each pixel as 3 bytes, one byte for the red component, one for green and one for blue. Each color in each pixel ranges from 0 (black) to 255 (the maximum color).

YCbCr is a different representation of color that separates the luminance (grayscale) (Y) component, which is the brightness, from the chroma components, which are the Cb (blue difference) and Cr (red difference). By separating the grayscale values from the color values, systems can selectively remove the color in a manner that is less visible to the human eye, which greatly improves data compression.

The Python code, minus the function you must fill-in, and two test cases are included in the assignment. See the attachments.

Conversion Formula:

As there are several conversion options, for this project you must convert an RGB pixel to a YCbCr pixel using the following formula for ITU-R BT.601 conversion [1].

```
// Standard BT.601 YCbCr conversion formula
 y = (char) round(0.299 * r + 0.587 * g + 0.114 * b);
cb = (char) round(128 - 0.168736 * r - 0.331264 * g + 0.5 * b);
cr = (char) round(128 + 0.5 * r - 0.418688 * g - 0.081312 * b);
```

C Kernel :

You will write your kernel in a separate file called “kernel.c”. It must have a function with the following signature:

```
void convert_to_YCrCb(unsigned char *rgb_pixels, unsigned char
*ycc_pixels, int width, int height)
```

The single dimensional input array `rgb_pixels` is passed with a pointer and is a total of width*height pixels, in row major order, and each pixel is 3 bytes. The output is passed in the `ycc_pixels` array, which has the same size as the `rgb_pixels` array. You can assume the input arrays are properly sized with the exception that your code should be robust to zero values. That is it should not crash if the width or height is zero, but can return an array of zeros for `ycc_pixels`.

To create a shared object (.so) for your C code, you will need to use the flags "-shared". You should also compile with the highest level of optimization, so add the -O3 flag to the compile line in your makefile as well. You will need to include the math library in your code to get the round() function to work.

Python Function:

The provided Python file has an empty function `convert_RGB_to_YCbCr(rgb_pixels, width, height)`. You must finish the code for this function. It should return a byte array with the YCbCr pixels from the converted image.

You may not use any libraries for the Python conversion function. It must be in pure Python. Note that Gradescope docker container may not contain many Python libraries, so do not use any beyond the ones in the Python script.

Handling numeric differences in the conversion:

Often in the case of writing C kernels, the results of floating point operations from one language do not exactly match the C version. Discrepancies can arise because of different rounding modes, differences when converting integers to floating point and back, and differences in the order of operations when evaluating expressions.

If any pixel differs by more than one value (e.g. abs(Python(YCbCr) - C_library(YCbCr)>1) you have done something wrong and the program will return an error.

Output Format:

The program should return the grayscale image resulting from the conversion and the time of both the Python and C conversion.

The program must output exactly 2 lines of the form:

Similarity Check: <number of identical bytes>, Different Bytes: <number of bytes different>

Converted <total_pixels> pixels with the C kernel in <C-kernel-time> seconds and with the Python code in <Python-conversion-time-in-seconds> "

The values to output are:

1. The *number of identical bytes*: The number of YCbCr bytes that are exactly the same from the Python and C versions of the conversion
2. The *number of bytes different:* The number of YCbCr bytes that are different from the Python and C versions with a maximum difference of 1.
3. *Total Pixels*: The total number of pixels in the image.

4. *C-Kernel-time*: The time to call the C Kernel and return the data, as a floating point number in seconds with at least 6 digits of precision.
5. *Python-conversion-time-in-seconds*: The time to convert the RGB pixels to YCbCr and return the result in the Python3 function `convert_RGB_to_YCbCr`

Example Usage:

convertRGB.py -i <input-filename> -o <output-filename>

Two image files will be provided, these are example outputs:

convertRGB.py -i testimage1.jpg -o testimage1_bw.jpg

convertRGB.py -i testimage2.jpg -o testimage2_bw.jpg

Submission Requirements:

If you are working in a group, you **must submit as a group in Gradescope**. See the link below for instructions.

Submit your C source code in a single `` `kernel.c` `` file along with a makefile called `` `makefile` `` that compiles `kernel.c` to a shared object (.so file) called `` `libkernel.so` ``. **You will receive a 0 if these instructions are not followed.**

You must also hand in a python3 file called “convertRGB.py” which completes the function ‘’ that we left blank.

Grading:

Your program will be tested against 10 test cases in increasing order of difficulty. Each test case is worth 10 points for a total of 100 points. Two test cases are included in the *example* section of this write-up.

You need to upload the two files `` `kernel.c` `` and `` `makefile` `` to Gradescope. Log into canvas and use the Gradescope tool on the left.

How to add group members in gradescope:
https://help.gradescope.com/article/m5qz2xsnjy-student-add-group-members

[1] https://en.wikipedia.org/wiki/YCbCr