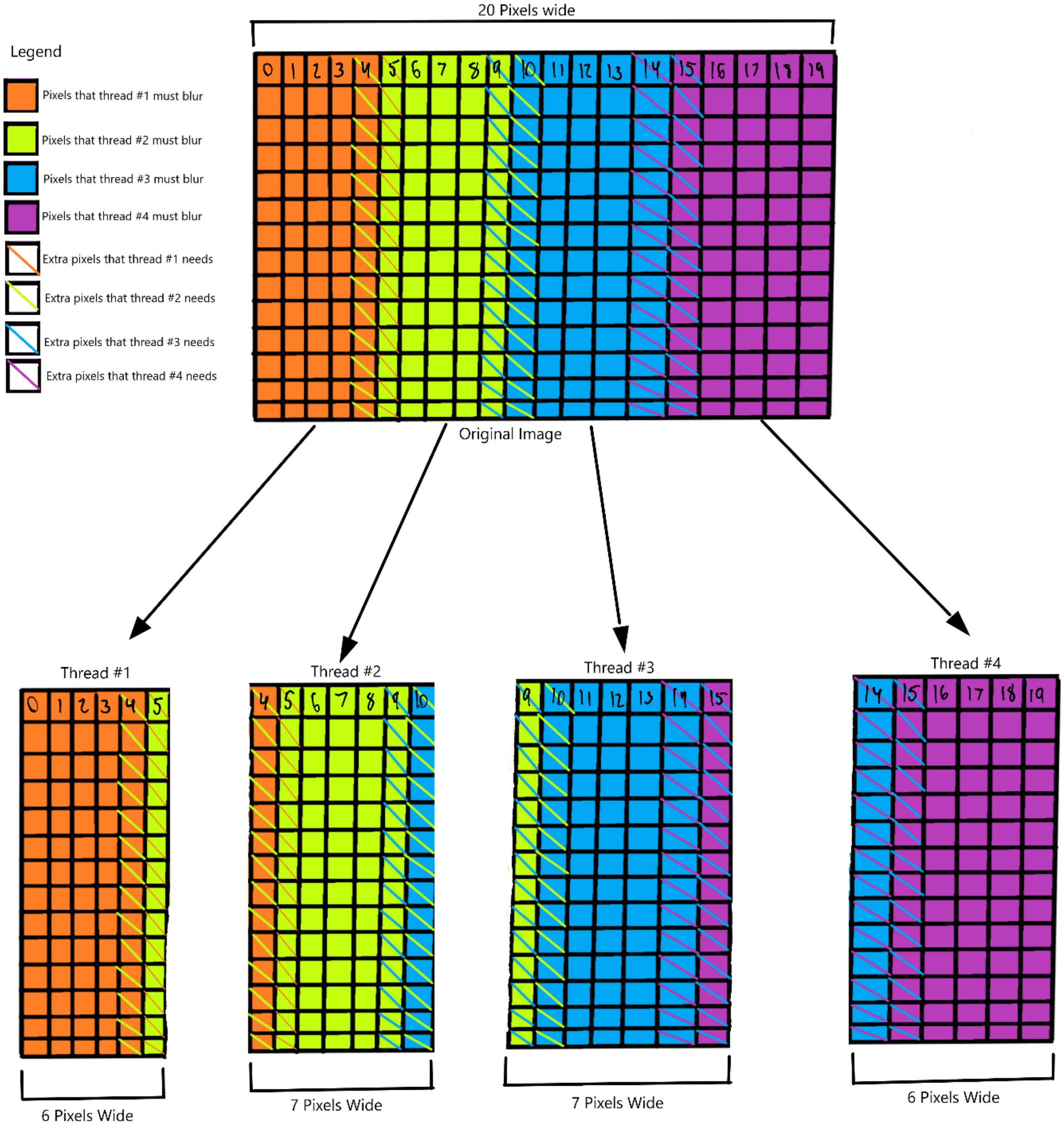
Assignment 6 Hint List

Hint 1: Blur Filter – Data splitting

To apply the box blur algorithm to an image using multiple threads, we need to divide the original image into smaller pieces and provide each thread with an independent allocation of pixels. However, the algorithm requires access to all surrounding pixels to blur a given pixel. This means that we must provide additional pixels to each thread (the assignment refers to these extra pixels as “padding”), even if it is not responsible for blurring them.

For example, if we split a pixel array that is 20 pixels wide among four threads, thread #1 will blur pixel columns 0-4, thread #2 will blur columns 5-9, and so on. However, for thread #1 to blur column 4, it requires access to the pixels in column 5. Thus, even though thread #1 is only responsible for five rows of pixels, it needs access to an array that is six pixels wide. It's important to note that not all threads require the same number of padding pixels. The threads that blur the left and right sides of the image only require an array 6 pixels wide, whereas the threads blurring the inner blocks require arrays that are 7 pixels wide.

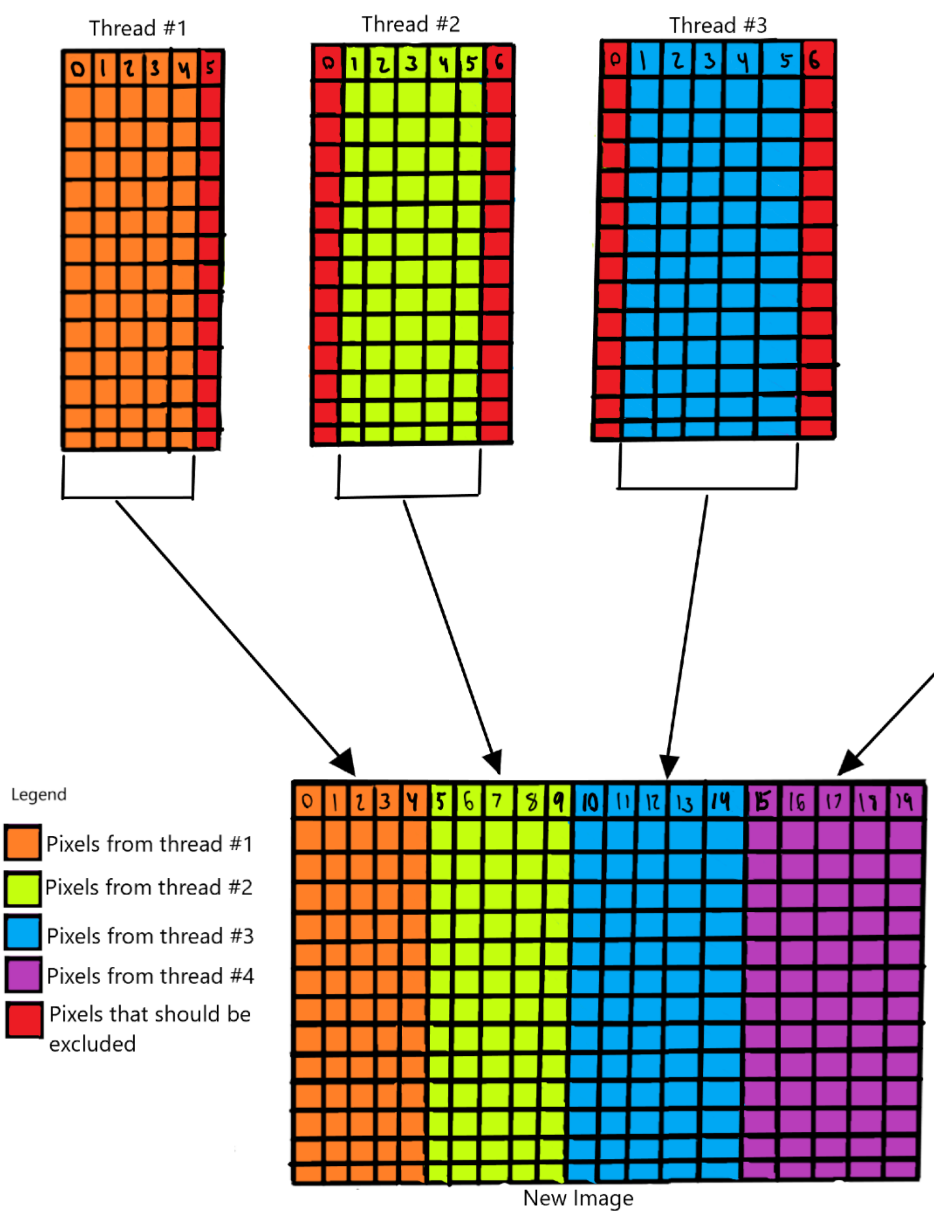
Blur Filter: Splitting an Image


20 Pixels wide
Legend
Pixels that thread #1 must blur
Pixels that thread #2 must blur
Pixels that thread #3 must blur
Pixels that thread #4 must blur
Extra pixels that thread #1 needs
Extra pixels that thread #2 needs
Extra pixels that thread #3 needs
Extra pixels that thread #4 needs
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
Original Image
Thread #1
0 1 2 3 4 5
6 Pixels Wide
Thread #2
4 5 6 7 8 9 10
7 Pixels Wide
Thread #3
9 10 11 12 13 14 15
7 Pixels Wide
Thread #4
14 15 16 17 18 19
6 Pixels Wide

Hint 2: Blur filter – Rejoining the image

After increasing the width of each thread, a new challenge arises when trying to merge the image back together after applying the blur filter. The initial image was 20 pixels wide, however, merging each thread's pixel arrays directly would result in a final image that is 26 pixels wide due to the added width. Therefore, it is necessary to exclude the padding pixel columns from each thread when rejoining the image.

Blur Filter: Rejoining an Image



Hint 3: Cheese filter – Yellow tint

This task may initially seem trivial if you were able to successfully implement the color shift function in module 3, but this task is slightly more complicated. There’s no fixed RBG shift that you can apply to an entire image to make it more yellow, you may need to calculate the RBG shift for each individual pixel. Remember, yellow is a combination of red and green, so to shift a single pixel towards yellow, typically you need to increase R and G values (and possibly decrease B). The challenge is to figure out how much to shift each value!

Hint 4: Cheese filter – Drawing circles

The key to this task is to think of your pixel arrays as a coordinate plane, where the rows are on the y axis and the columns are on the x axis. Then try to think about how you would draw a circle on a coordinate plane. Every circle can be defined by three attributes: the x and y coordinates of its center, and its radius. Every point within the radius is a part of the circle, and every part outside of the circle is not.

The challenging part of this task is to draw the circles using threads. Since each thread has its own pixel array, the x coordinates of each point will not match the corresponding x coordinate of the original array, which may make it difficult to calculate the distance to the center of a given circle. This means that each thread will need to “know” which portion of the original image it is operating on, so that you can apply the correct offset to each point’s x coordinate.

Hint 5: CLion CMakeList.txt

The homework PDF states that CLion automatically links math.h, but if you have any issues with CLion not recognizing math.h, you can add

$$target\_link\_libraries(\{enter\ target\ name\ here\}\ m)$$

to your CMakeList.txt file to link math.h manually.