Assignment 3  
Mohamed Shafiq Bin Roslan | 6098253

### Task 1

We were to do a Gaussian Blur with the window and the double pass method. For part A, after selecting the device and creating the command queue, I read the image in that is to be blurred, and send it into the input buffer. The output buffer is to read the changed image and the choice is from the user selecting the window size.

In the kernel, there are blurring constants that are used to set the blur of each pixel and the pixels surrounding it. The queued buffers are sent to part\_a where each pixel is slowly changed together with pixels surrounding it depending on which window is chosen. After that is done, the image is then written to the output buffer to be read later and to save the new image in a bmp file.

For part B it is similar to part A, the difference being that the images are sent twice to the kernels to be blurred horizontally and vertically. For the first pass, the image is sent over to the kernel, and blurred only in the vertical manner, it is then sent back to the main program where the output from the first pass is then sent to the same kernel to be blurred horizontally. The kernel takes in the information for which pass it is currently in so as to not accidentally do the same directional blurring again. The image is then sent back to the main program for reading and printing.

While doing part C, I found that the 2-pass runs faster than the 3x3/5x5/7x7 method. I think this is due to the work being halved by the two kernels instead of just one kernel doing everything at once. GPU also runs faster than CPU for both methods and it shows that the kernels are running in parallel.

### Task 2

We were to change the given image to a greyscale image and also calculate the average luminance of the picture.

When the image is sent to kernel, for each pixel in the image, the pixel is converted into RBGA numbers and then converted into greyscale with the given formula in the question. It is then converted back to its original form with the numbers adjusted to greyscale. It is then sent back to the main program to be read and printed.

On the main program, the average luminance is calculated for part B. The loop calculates for each pixel in the image, it moves through the RGBA stored in each pixel and extracts and calculates each colour and the RGB is stored into the total luminance to be divided and shown.

For part C, the calculations are sent into two kernels to divide the workload of calculating the average luminance. The kernels are each storing the sum of each RGBA luminance in the picture. In the second kernel it continues to do that as well and at the end it combines all four RGBA sums into one grand total to be divided to show the average later.

### Task 3

For task 3, we are to combine what we learn above into one cohesive program that takes in a picture, calculates the average luminance, using the average luminance darkens everything that is below the average luminance, use a double pass on the darken image, and then combine it with the original image to create a bloom effect.

Firstly, I used the method from 2B to calculate the average luminance and return the total luminance to the host program and calculate the average. I then ask for a user input if they want to use a different value instead to set threshold values to darken the picture. Using the values received above, the image is sent to the kernel and in the kernel, everything that is below the threshold is set to black. The image is then returned and then sent to the double pass to be blurred. User is also asked for a choice between 3, 5 and 7 for their blurring window. Like in task 1, the image goes through a double pass and then returns with the image being horizontally and vertically blurred. Finally, the original image and the edited image is sent to the kernel to be combined. Both images are read in the kernel and then added to each other to produce one image.