Name:\_\_Xiang, Xin\_\_ Date: \_\_\_\_ \_May 17, 2020\_ \_ \_

Section: \_ 005\_ \_ \_ Github Username: \_\_FreyaXiang\_ \_

**Lab 6**

Total in points (100 points total):

Professor’s Comments:

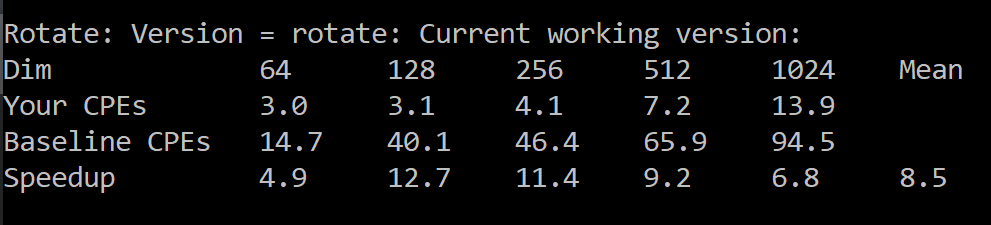
Honor Pledge: I have neither given nor received aid on this assignment.

Signature: Xin Xiang

7.

To better optimize the rotate() function, I rotated the image by continuously modifying the position of the two pointers (\*src/\*dst). Because the dimension of the matrix is always a multiple of 32 and the matrix is a square matrix, I used fewer loops (two loops instead of four loops). For the inner loop, I write everything explicitly for 32 times instead of using more loops.

Performance:

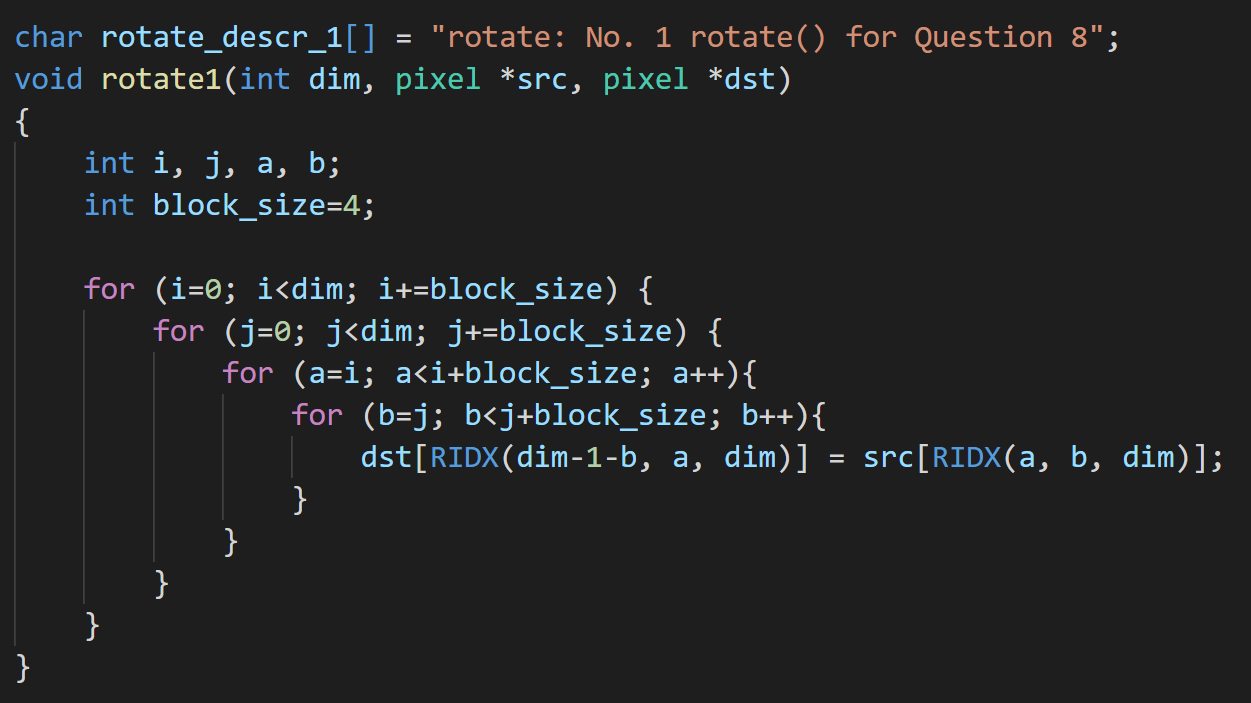


The speedup mean is 8.5 in this case. This greatly improves the efficiency of the program.

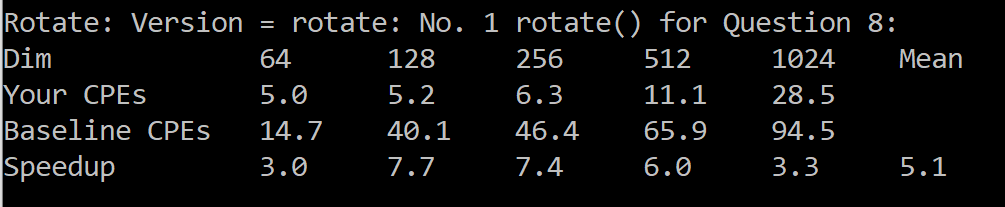
8.

I write two more rotate() functions using the blocking method with different blocking sizes.

The block size of the first one is 4.

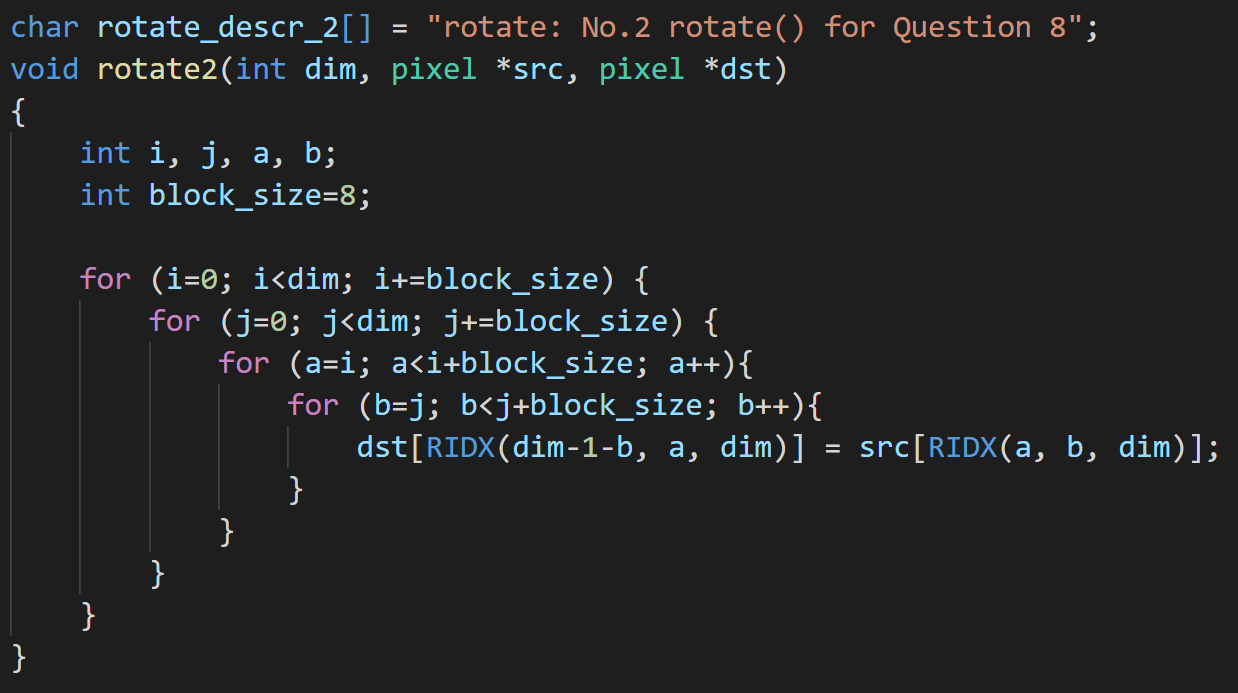


Performance:

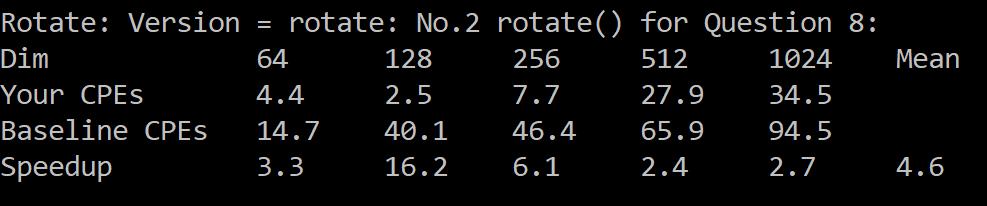


The speedup mean is 5.1.

The block size of the second one is 8.



Performance:



The speedup mean is 4.6.

For each function, I used 4 loops. The outer two are used to iterate across each block. The two inner loops are used to iterate through each element in the block. From these two functions, we see that the function with block size 4 is more efficient than the other one. For block size 4, we are dividing the matrix into smaller parts.

But the speedup of Question 7 is 8.5, which is the greatest one among these three. This is because I used fewer loops in question 7.

9.

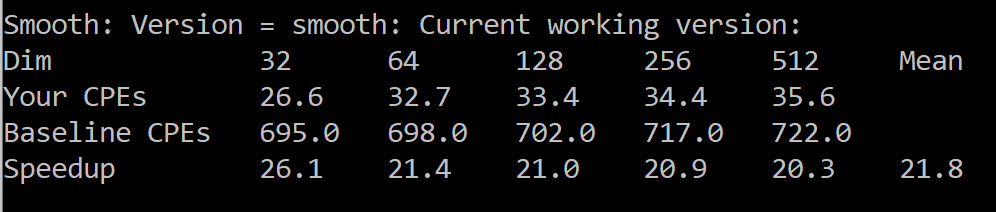
To reduce cache miss/loops, I write this function in three steps.

First, compute the RGB values for the four corners separately.

Then, compute the RGB values for the four edges by using separate loops.

Finally, for the remaining pixels in the middle, I used nested loops to compute the RGB values for each pixel.

Performance:



The speedup mean is 21.8 in this case. This also greatly improves the efficiency of the program.