CMOR 420/520 Fall 2024

C assignment:

1. Create a MyMatrix struct which holds an (m-by-n) dense matrix
    1. Fields should be “int num_rows, num_cols” and “double ** data”
    2. For data, use Format 3 in class (contiguous memory and the pointer reassignment trick)
    3. Create a function “initial_MyMatrix(int m, int n)
2. Implement a “matmul” routine between two MyMatrix instances.
    1. It should return an “int” error code, where “0” is no error, and “1” indicates size compatibility error.
    2. The output should be passed in as a MyMatrix argument
    3. Test for correctness: compute [1 2 3 4; 5 6 7 8; 9 10 11 12] * [1 2 3; 4 5 6; 7 8 9; 10 11 12]
        1. Answer should be
            1. 70 80 90
            2. 158 184 210
            3. 246 288 330
        2. Compute and print out the maximum absolute difference between the exact and computed matrix
        3. Put this in “test_matmul.c”
3. Compare runtimes (use the minimum time over 25 samples) timing
    1. Compute runtimes for two square matrices with sizes from 10 to 512.
    2. Put the timing code in “time_matmul.c”. This code should output the timings. Run this for both gcc -O0 and gcc -O3
    3. Plot the runtimes for both gcc -O0 and gcc -O3 using your preferred plotting software (Matlab, Python, Julia, Excel, etc) and include them in your writeup.
4. Code organization and compilation
    1. Make sure your code is readable: at minimum, indent your code properly

and provide comments when necessary.
2. All struct definitions and function prototypes should be placed in “include/MyMatrix.h”
3. All function definitions should be placed in “src/MyMatrix.c”

5. Written questions:
    1. Explain whether it would or would not be possible to implement a contiguous-in-memory column major format in C using the same pointer reassignment trick as Format 3.
    2. Suppose you wished to write functions to append one MyMatrix to an existing MyMatrix either vertically or horizontally. Which one would be easier and why? Be specific.
6. Grad students:
    1. Implement a “reshape(int new_m, int new_n, MyMatrix * A)” routine. This should:
        1. Modify the input MyMatrix *
        2. Keep the underlying data layout
        3. Check compatibility of dimensions
        4. Test correctness of your implementation in “test_reshape.c”. Provide build instructions for this in your writeup.