-0.7	-0.4	0.6	1.0	-0.6	-0.5	0.3	-0.8	0.9	-0.8
-0.7	-0.1	-0.6	0.1	0.9	0.1	-0.9	0.6	0.2	0.5
1.0	-0.4	0.5	0.7	-0.8	0.9	-0.8	0.8	-0.6	0.8
0.8	0.6	-0.5	0.2	0.1	-0.9	-1.0	0.2	0.3	0.5
0.0	0.4	0.8	-0.7	-0.7	0.9	-1.0	0.1	0.7	-0.9
-0.5	-0.3	0.7	-0.2	-0.7	0.0	0.0	-0.7	-0.1	0.9
-0.4	0.3	-0.8	-0.4	0.1	0.7	0.2	1.0	-0.9	0.0
0.5	0.9	0.3	0.1	0.5	-0.4	0.8	-0.8	0.9	1.0
0.5	0.2	-1.0	0.4	0.2	1.0	-1.0	0.9	1.0	-0.3
0.8	-0.4	0.2	-0.9	0.5	-0.7	0.1	0.2	0.7	-0.7

24-780 B—Engineering Computation

Assigned: Wed. Nov. 30, 2022
Due: Wed. Nov. 30, 2022, 11:59pm
(Will do all the work in the lecture on Wed., Nov. 30)

Problem Set 10: Matrices

This assignment will take the form on an in-lecture exercise, so whatever you have done at the end of the lecture, just submit. The goal here is to get you some exposure, but with the understanding that most of your mind is now on the final project.

Task 1 (Get the start-up code)

Yes. Do that, get the code and we'll go over it some

Task 2 (Do the little things)

As part of understanding how everything works, let's fill out the code for some of the functions that are currently mostly empty bodies.

Task 3 (Matrix multiply)

The matrix multiply function is a little tricky and can be quite computationally intensive.

Task 4 (Matrix multiply using multi-threading)

Matrix multiplication is rather slow. Implement the following function so that we can use the full power of 21st century computing. In addition to the following functions, you need to come up with a mechanism for controlling the threads and avoiding the race condition.

```
// similar to matrixMultiply, but uses multi-threading
void matrixMultiplyParallel(Matrix2D<T, NC, NR>& otherMatrix,
          Matrix2D <T, NR, NR>& resultMatrix);

// thread entry static function
public: static void threadEntry(Matrix2D<T, NR, NC>* thisPtr);
```

Deliverables

All files you create, very appropriately named and zipped together which at a minimum include:

ps11_matrix_andrewID.h << contains declaration and bodies of the classes you develop

Upload the zip file to the class Canvas page before the deadline (Wed., Nov. 30, 11:59pm).

Learning Objectives

Deep Copy

Multi-threading

Template classes.

Array manipulation.

Computational programming using matrices.

Using tester programs for class debugging.

Searching references (online or textbook) for C++ library functions.