-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: Mon. Nov. 15, 2021
Due: Tues. Nov. 23, 2021, 11:59pm
(But will do a lot of the work on Mon., Nov. 15)

Problem Set 9: Matrices

This assignment will not require any graphics or sound, so keep working hard on your team project. Now, we will improve our skills in good-ol' number crunching, namely playing with matrices. You are required to create the following:

Matrix2D – a template class that takes three template parameters:

- class T – the datatype of the elements in the matrix (some kind of numeric type)
- const int NR – the number of rows in the matrix
- const int NC – the number of columns in the matrix

The class will use a **private** array member variable called *content* which will hold the data. Note that the array must be **dynamically allocated** so that we can hold really big arrays (5000x5000 or more) without risk of overflowing the memory stack.

The class will have the following functions (all public):

```
// Class constructor and destructor
Matrix2D();
~Matrix2D();

// Sets value at given location, rows and columns are numbered 1 to whatever
// Use the following if row or col are out-of-range
//      ->> throw std::out_of_range("Bad matrix operation");
void set(int row, int col, const T &incoming);

// Gets value at given location, throwing error similar to set()
const T &value(int row, int col) const;

// Loads matrix data from the file given
void readFile(const std::string& fileName);

// Prints the whole array to output, defaulting to cout.
// If positive limit is given, only the first printLimit rows and columns
// are printed rather than whole matrix (useful for checking big matrices).
void print(std::ostream& output = std::cout, int printLimit = -1) const;

// copy constructor
Matrix2D(const Matrix2D<T, NR, NC>& original);

// assignment operator
Matrix2D<T, NR, NC>& operator=(const Matrix2D<T, NR, NC>& original);

// Transposes the matrix in place.
void transpose();

// multiply the matrix by a scalar
void scalarMultiply(const T& factor);
```

```
// Multiplies the matrix with otherMatrix and stores the result in the
// resultMatrix. Note that otherMatrix is the right-hand operand (since matrix
// multiplication is NOT commutative)
void matrixMultiply(Matrix2D<T, NC, NR>& otherMatrix,
        Matrix2D <T, NR, NC>& resultMatrix);
```

You can use Matlab to check matrix calculation or use the attached Excel file. .
(Yes, Excel can do matrices ☺)

In addition to creating the class for Matrix2D, also play around with the main() function (given in a separate file) to test the functions of the class. Make use of a timer to check how long the two multiplication operations take using code similar to thefollowing:

```
auto begin = std::chrono::high_resolution_clock::now();
double timeElapsed = chrono::duration_cast<chrono::nanoseconds>
       (std::chrono::high_resolution_clock::now() - begin).count() / 1e9;
```

Deliverables

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

ps09_matrix2D_andrewID.h << contains declaration and bodies of the classes you develop

ps09_matrix_checker_andrewID.cpp << contains a series of simple tests for your class functions

Upload the zip file to the class Canvas page before the deadline (Tuesday, Nov.23, 11:59pm).

Learning Objectives

Template classes.

Array manipulation.

Computational programming using matrices.

Using tester programs for class debugging.

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