Custom Matrix Arithmetic

1 - Introduction

In this project, you will be exploring the implementation of matrix arithmetic. There are third-party libraries that provide matrix operations, but for this project, you will be implementing your own C++ class and overloading some matrix operators. The member functions/operators you will be required to implement are as follows:

- *Constructor*
- *Copy Constructor*
- *Destructor*
- *Matrix Output Operator (<<)*
- *Matrix Input Operator (>>)*
- *Matrix Addition Operator (+)*
- *Matrix Subtraction Operator (-)*
- *Matrix Addition-and-Assignment (In-place Addition) Operator (+=)*
- *Matrix Subtraction-and-Assignment (In-place Subtraction) Operator (-=)*
- *Matrix Multiplication Operator (*)*

2 - Background

The goal of this project is to implement various matrix operators using a custom Matrix class. Therefore, it is important to establish how these operations are to be performed. All matrix addition/subtraction operations are to be performed using element-wise addition/subtraction. Consider the addition of two matrices **A** and **B** where the result is **C**. Both **A** and **B** are n x m matrices.

A + **B** = **C**

Element-wise addition will thus yield:

$$c_{i,j} = a_{i,j} + b_{i,j}$$

i = 0 to n - 1

j = 0 to m - 1

n = number of rows

m = number of columns

Consider the multiplication of matrices **A** and **B** (in that order) where the result is **C**.

A * **B** = **C**

A is an n x m matrix

B is an m x p matrix

A generic equation to perform matrix multiplication is:

$$c_{i,j} = \sum_{k=0}^{m-1} a_{i,k} b_{k,j}$$

i = 0 to n - 1

j = 0 to p - 1

3 - Tasks

Implement the **constructor, copy constructor, destructor, custom <<, >>, +, -, +=, -=, and * operators** for matrix arithmetic in the Matrix.cpp file. Check the comments starting with TODO for more details.

- You can find information about the members of the Matrix class in the Matrix.h file.
- The main() function in main.cpp receives the operation type (matrix addition, subtraction, in-place addition, in-place subtraction, or multiplication). Each test function in main.cpp receives the number of rows, the number of columns, and matrix element values and calls the overloaded operators. There is nothing to do with the main.cpp file, but check how it works.
- **The operator+(), operator-(), and operator*() member functions must return a new Matrix object with the result of the operation.**
- **The operator+=() and operator-=() must return a reference to the current object. You may return *this.**
    - Q) Why do we need to return the current object when implementing the operator+=() and operator-=()?
    - A) Actually, the member values are updated in place before a return statement, which means that the code without a return statement will pass the test using run.sh. The reason why those overloaded operators return the current object is to support consecutive operations on the same code line, such as "a += (a += b)" and “cout << (a += b)”. Although this case is not tested, it is common for overloaded operators to return the current object for this purpose.

- The assert() function is used to check for specific conditions. If the condition within () is true, nothing happens. Otherwise, the program terminates and an error message is printed. See the example in the operator+() member function.
- For alignment purposes, use setw() and the keyword “fixed”. Here is an example:

```
output << setw(mat.OUTPUT_WIDTH) << fixed << your_value;
```

- Your results will be written in the generated result.txt file. You should open this file to check if your results are correct.

4 - Compile and Test

Type the following commands on Terminal.

```
g++ -o main *.cpp *.h
./main
```

Inputs:

- Operation
  - 0: matrix addition
  - 1: matrix subtraction
  - 2: matrix in-place addition
  - 3: matrix in-place subtraction
  - 4: matrix multiplication
- The number of rows and columns of the first input matrix
- The (double-typed) element values of the first input matrix (row by row)
- The number of rows and columns of the second input matrix
- The (double-typed) element values of the second input matrix (row by row)

```
#example: matrix addition
0 # matrix addition
2 3 # row and column
1.0 2.0 3.0 # element values in the first row
4.0 5.0 6.0 # element values in the second row
2 3 # row and column
7.0 8.0 9.0 # element values in the first row
0.0 1.0 2.0 # element values in the second row
#expected result in result.txt
#8.00 10.00 12.00
#4.00  6.00  8.00
```

To check all together, you can use the following commands:

```
chmod 777 run.sh
./run.sh
```

You must see the following messages:

```
ADD - PASS
SUB - PASS
ADDINPLACE - PASS
SUBINPLACE - PASS
MUL - PASS
```

5 - Submit

- Please upload your **Matrix.cpp** file to myCourses > Assignments > Project2.
- Do not submit *.docx, *.pdf, *.txt, or *.zip file.
- Do not change other files.
- Only one student from each team can submit.

6 - Grading

- [90 pts] - Functions to be implemented
    - [5 pts] constructor
    - [10 pts] copy constructor
    - [5 pts] destructor
    - [10 pts] operator<<()
    - [10 pts] operator>>()
    - [10 pts] operator+()
    - [10 pts] operator-()
    - [10 pts] operator+=()
    - [10 pts] operator-=()
    - [10 pts] operator*()
- [10 pts] - Program execution
    - **The program execution portion of the grade is ALL or nothing.**