Programming Assignment 1

Problem Specification

Create a **Matrix** class, which stores an $m \times n$ matrix of *template type* values, with the following specifications below. Make sure that all functions, overloaded operators, and constructors are *public*.

1. Must have two constructors: one for *initialization* and one for *copy*. More details are described below.

   (a). The *initialization* constructor takes in two *unsigned int* constants m and n, and a constant *template type* array x with mn elements. The $m \times n$ matrix is initialized using x starting from the first row, filled from left to right, until the last row. You may assume that x contains exactly mn values.

   (b). The *copy* constructor takes a constant reference to a Matrix object, and copies each element onto the new matrix.

2. Must support the = operator, which is similar to the *copy* constructor.

3. Must support the << operator for the std::cout object. Display through the std::cout object should print all the elements of a Matrix object in a comprehensible way, e.g. each row is separated by a newline.

4. Must have an access function *at()*. More specifically, $at(i, j)$ returns a reference to the element at the ith row and the jth column of the matrix. This access function can be used to either get or set the value of an element of the matrix.

5. Must support the matrix operations listed below. You may assume that matrix operands are of the same Matrix *template type*.

   (a). *Addition* using the + operator returns a Matrix object, where each element is derived from the sum of the elements of the two matrices.
   More formally, if A, B, C are $m \times n$ matrices and $C = A + B$, then:

   $$C_{i,j} = A_{i,j} + B_{i,j}$$

   for all $1 \leq i \leq m$ and $1 \leq j \leq n$. Here $C_{i,j}$ denotes the ith row and the jth column of matrix C.

   (b). *Subtraction* using the − operator returns a Matrix object, where each element is derived from the difference of the elements of the two matrices.
   More formally, if A, B, C are $m \times n$ matrices and $C = A - B$, then:

   $$C_{i,j} = A_{i,j} - B_{i,j}$$

   for all $1 \leq i \leq m$ and $1 \leq j \leq n$.

   (c). *Scalar Multiplication* using the $*$ operator takes in a *template type* as the multiplier, and returns a Matrix object with the original matrix scaled in proportion to the value provided.
   More formally, if M, P are $m \times n$ matrices, x is a real number, and $P = M \cdot x$, then:

   $$P_{i,j} = M_{i,j} \cdot x$$

   for all $1 \leq i \leq m$ and $1 \leq j \leq n$.

   (d). *Matrix Multiplication* using the $*$ operator takes another Matrix object as the multiplier, and returns a Matrix object which is the result of multiplying the two matrices.
   More formally, if A is an $m \times p$ matrix, B is a $p \times n$ matrix, C is an $m \times n$ matrix, and $C = A \cdot B$, then:

   $$C_{i,j} = \sum_{k=1}^{p} A_{i,k} \cdot B_{k,j}$$

   for all $1 \leq i \leq m$ and $1 \leq j \leq n$.

6. Must provide the static initializer functions listed below. *Note: Static initializer functions are declared using the static keyword and can be used to initialize an object without a need for an object instance to invoke it, e.g.* $Matrix < int >:: eye(3)$.

   (a). Identity matrix using the *eye()*. More specifically, eye(n) returns an $n \times n$ Matrix object whose diagonal elements are 1 and off-diagonal elements are 0.

   (b). Zero matrix using *zeros()*. More specifically, *zeros(m, n)* returns an $m \times n$ Matrix object whose elements are all 0.

   (c). One matrix using *ones()*. More specifically, *ones(m, n)* returns an $m \times n$ Matrix object whose elements are all 1.

Bonus

Support the [][] operator, which is similar to the *at()* access function. The code snippet below should work.

```
// mat is declared as a 2x2 Matrix<double> object
// ...
mat[1][2] = 2.5;
std::cout << mat[1][1] << std::endl;
```

Notes

1. If you have a question involving source code, please do post the code snippet and/or error messages. Always resolve compile-time errors in the order they appear in your code. If you think that the code snippet might be too revealing of your work, post it as a private question first and I'll decide if said code may be made public to your classmates.

2. Make sure that you are using a g++-8 compiler which is either from GCC 8.1 or GCC 8.2 so that you will not have issues during checking. If you want to use a new C++ specification kindly comment what compile command you are using as a comment at the beginning of your source code.

3. You may use any container in the standard library, pointers, or smart pointers to store the elements of the **Matrix** class.

4. All unspecified parts are made open to interpretation. If you really want to make sure, just post a question at Piazza.

5. Provide a test for **all** functionalities required in the specifications in the main function.

Submission

1. Soft deadline is set to 9:00 PM of 18 February 2019.

2. Hard deadline is set to 9:00 PM of 26 February 2019.