CS/CE 224/272 - Object Oriented Programming and Design Methodologies: Assignment #3

Fall Semester 2024

Due on November 23, 2024, 11.59pm

Student Name, ID, Lecture Section

Instructions

1. This homework consists of one large programming exercise.
2. Please submit the solution to this Assignment in the corresponding .cpp files that will be made available to you corresponding to each class as follows:
   (a) Matrix.cpp
   (b) SquareMatrix.cpp
   (c) UpperTriangularMatrix.cpp
   (d) LowerTriangularMatrix.cpp
   (e) Diagonal.cpp
3. The header files (*.hpp) and the driver file (main.cpp) for testing will be provided to you. You do not have to modify or submit these files.
4. Make sure your name, ID and section are written as a comment in the source code.
5. You can join the assignment repository by clicking on the link provided to you.
6. Submission will be via GitHub by uploading the corresponding repository.
7. No extensions will be given for Assignment Submission.
8. While collaboration and discussion are encouraged, this assignment is strictly individual. Plagiarism and copying, whether partial or complete will result in a *zero* grade for both students, whose assignments are found to be similar. Furthermore, the plagiarism may also be reported to the University Conduct Committee.

Problem 1

(100 points) [**Matrix Algebra System**]
Implement a comprehensive Matrix Algebra System that operates over the real number field $\mathbb{R}$. This matrix algebra system should support the following matrix types and operations:

Matrix Types:

1. General Matrix: Matrix A with m rows and n columns, where m and n are positive integers.
2. Square Matrix: Matrix S with n rows and n columns, where n is a positive integer.
3. Lower Triangular: Matrix L with n rows and n column where $n > 0$ and $l_{ij} = 0$ for $i < j$.
   Note that this is a square matrix where all elements *above* the main diagonal are zero.
4. Upper Triangular: Matrix U with n rows and n columns where $n > 0$ and $u_{ij} = 0$ for $i < j$.
   Note that this is a square matrix where all elements *below* the main diagonal are zero.
5. Diagonal Matrix: Matrix D with n rows and n columns where $n > 0$ and $d_{ij} = 0$ for $i \neq j$.
   Note that this is a square matrix where all elements *not* on the main diagonal are zero.

Inheritance Hierarchy:
In terms of Object Oriented Programming, an inheritance hierarchy of these matrices is shown as follows:

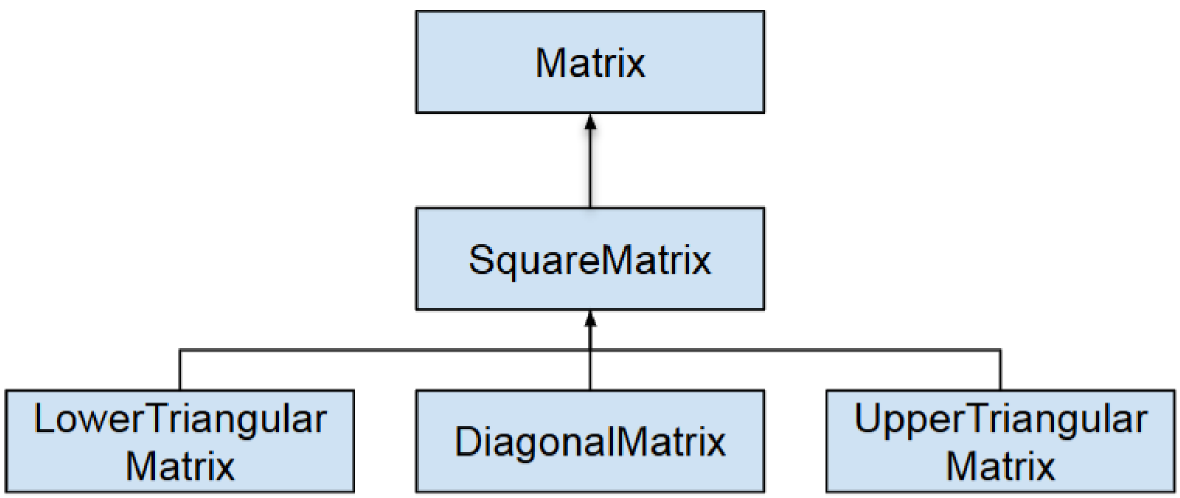

Figure 1: Inheritance hierarchy of matrix classes

Operations
For compatible matrices X, Y, where X is $m \times n$ matrix and Y is $p \times q$ matrix, the system should support the following operations:

1. Addition: $X + Y$, where $m = p$ and $n = q$.
2. Subtraction: $X - Y$, where $m = p$ and $n = q$.
3. Multiplication: $X \times Y$, where $n = p$. The resulting matrix has dimensions $m \times q$.
4. Comparison: $X == Y$, where $m = p$ and $n = q$.

Problem 1 continued on next page. . .

Implementation Requirements:
The system shall utilize C++ object-oriented features as follows:

- **Inheritance** hierarchy reflecting mathematical subset relationships,
- **Polymorphic operations** corresponding to matrix algebra,
- **Operator overloading** for natural mathematical notation,
- **Memory-efficient storage** exploiting matrix structure.

All operations must preserve the mathematical properties of each matrix type and maintain computational efficiency through specialized implementations.

Memory Layout:
For all matrix classes in this system, elements are stored in **row-major order** in a single `std::vector<double>` (vector of doubles). They are initialized with zero values to the required size.

Details of Classes

Matrix

Definition 1 (Matrix)**.** *Let $A \in \mathbb{R}^{m \times n}$ be a matrix with m rows and n columns:*

$$A = [a_{ij}]_{m \times n} = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,n-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m-1,0} & a_{m-1,1} & \cdots & a_{m-1,n-1} \end{bmatrix}$$

The matrix A is represented as a one-dimensional vector of size $m \cdot n$:

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ a_{1,0} & a_{1,1} & a_{1,2} \end{bmatrix}$$

```
std::vector<double> vec = [a[0][0], a[0][1], a[0][2], a[1][0], a[1][1], a[1][2]]
```

Properties:

- There are m rows and n columns.
- The total number of elements is $m \cdot n$.
- The index of element a_{ij} is $i \cdot n + j$ in the vector.

$$a_{ij} = \texttt{vec}[i \cdot n + j]$$

Required Functions: You have to implement the following functions: **Methods**:

Default constructor initializes an empty matrix:

```
Matrix() = default;
```

Parameterized constructor initializes a matrix with given dimensions:

```
Matrix(const int rows, const int cols);
```

Copy constructor creates a deep copy of another matrix:

```
Matrix(Matrix const& other);
```

Destructor releases memory allocated for matrix elements:

```
~Matrix();
```

Matrix addition, subtraction, and multiplication operators:

```
Matrix operator+(const Matrix& other) const; // Already made available
Matrix operator-(const Matrix& other) const;
Matrix operator*(const Matrix& other) const;
bool operator==(const Matrix& other) const;
```

Accessors for matrix dimensions:

```
virtual double getElement(const int row, const int col) const;
virtual void setElement(const int row, const int col, const double value);

int getRows() const;
int getCols() const;
```

Problem 1 continued on next page...

Square Matrix

Definition 2 (Square Matrix)**.** *A Square Matrix S is a matrix where $m = n$:*

$$S = [s_{ij}]_{n \times n} = \begin{bmatrix} s_{0,0} & s_{0,1} & \cdots & s_{0,n-1} \\ s_{1,0} & s_{1,1} & \cdots & s_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n-1,0} & s_{n-1,1} & \cdots & s_{n-1,n-1} \end{bmatrix}$$

The square matrix S is represented as a one-dimensional vector of size n^2:

$$S = \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} \\ s_{1,0} & s_{1,1} & s_{1,2} \\ s_{2,0} & s_{2,1} & s_{2,2} \end{bmatrix}$$

```
std::vector<double> vec = [s[0][0], s[0][1], s[0][2], s[1][0], s[1][1], s[1][2], s[2][0],
                           s[2][1], s[2][2]]
```

Properties:
- There are n rows and n columns.
- The total number of elements is n^2.
- It inherits from the Matrix class.
- The index of element s_{ij} is $i \cdot n + j$ in the vector.

$$s_{ij} = \texttt{vec}[i \cdot n + j]$$

Methods:
Constructor initializes a square matrix with given size:

```cpp
SquareMatrix(const int size);
```

Copy constructor creates a deep copy of another square matrix:

```cpp
SquareMatrix(SquareMatrix const& other);
```

Matrix addition, subtraction, and multiplication operators:

```cpp
SquareMatrix operator+(const SquareMatrix& other) const;
SquareMatrix operator-(const SquareMatrix& other) const;
SquareMatrix operator*(const SquareMatrix& other) const; // Already made available
bool operator==(const SquareMatrix& other) const;
```

Accessors for square matrix elements:

```cpp
double getElement(const int row, const int col) const;
void setElement(const int row, const int col, const double value);
```

Lower Triangular Matrix

Definition 3 (Lower Triangular Matrix)**.** *A Lower Triangular Matrix L is a square matrix where all elements above the main diagonal are zero:*

$$L = [l_{ij}]_{n\times n} = \begin{bmatrix} l_{0,0} & 0 & 0 & \cdots & 0 & 0 \\ l_{1,0} & l_{1,1} & 0 & \cdots & 0 & 0 \\ l_{2,0} & l_{2,1} & l_{2,2} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ l_{n-2,0} & l_{n-2,1} & l_{n-2,2} & \cdots & l_{n-2,n-2} & 0 \\ l_{n-1,0} & l_{n-1,1} & l_{n-1,2} & \cdots & l_{n-1,n-2} & l_{n-1,n-1} \end{bmatrix}$$

The lower triangular matrix L is represented as a one-dimensional vector of size $\frac{n(n+1)}{2}$:

$$L = \begin{bmatrix} l_{0,0} & 0 & 0 \\ l_{1,0} & l_{1,1} & 0 \\ l_{2,0} & l_{2,1} & l_{2,2} \end{bmatrix}$$

```
std::vector<double> vec = [l[0][0], l[1][0], l[1][1], l[2][0], l[2][1], l[2][2]]
```

Properties:

- There are n rows and n columns.
- The total number of elements is $\frac{n(n+1)}{2}$.
- Storage optimization by only storing values below the main diagonal (including the diagonal).
- It inherits from the SquareMatrix class.
- The index of element l_{ij} is $\frac{i(i+1)}{2} + j$ in the vector.

$$l_{ij} = \begin{cases} \texttt{vec}\left[\frac{i(i+1)}{2} + j\right] & \text{if } i \geq j \\ 0 & \text{otherwise} \end{cases}$$

Methods:
Constructor initializes a lower triangular matrix with given size:

```
(const int size);
```

Copy constructor creates a deep copy of another lower triangular matrix:

```
(LowerTriangularMatrix const& other);
```

Matrix addition, subtraction, and multiplication operators:

```
LowerTriangularMatrix operator+(const LowerTriangularMatrix& other) const;
LowerTriangularMatrix operator-(const LowerTriangularMatrix& other) const;
LowerTriangularMatrix operator*(const LowerTriangularMatrix& other) const;
bool operator==(const LowerTriangularMatrix& other) const;
```

Accessors for lower triangular matrix elements:

```
double getElement(const int row, const int col) const;
void setElement(const int row, const int col, const double value); // Already made available
```

Problem 1 continued on next page...

Upper Triangular Matrix

Definition 4 (Upper Triangular Matrix)**.** *An Upper Triangular Matrix U is a square matrix where all elements below the main diagonal are zero:*

$$U = [u_{ij}]_{n\times n} = \begin{bmatrix} u_{0,0} & 0 & 0 & \cdots & 0 & 0 \\ u_{1,0} & u_{1,1} & 0 & \cdots & 0 & 0 \\ u_{2,0} & u_{2,1} & u_{2,2} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ u_{n-2,0} & u_{n-2,1} & u_{n-2,2} & \cdots & u_{n-2,n-2} & 0 \\ u_{n-1,0} & u_{n-1,1} & u_{n-1,2} & \cdots & u_{n-1,n-2} & u_{n-1,n-1} \end{bmatrix}$$

The upper triangular matrix U is represented as a one-dimensional vector of size $\frac{n(n+1)}{2}$:

$$U = \begin{bmatrix} u_{0,0} & u_{0,1} & u_{0,2} \\ 0 & u_{1,1} & u_{1,2} \\ 0 & 0 & u_{2,2} \end{bmatrix}$$

```
std::vector<double> vec = [u[0][0], u[0][1], u[0][2], u[1][1], u[1][2], u[2][2]]
```

Properties:

- There are n rows and n columns.
- The total number of elements is $\frac{n(n+1)}{2}$.
- Storage optimization by only storing values above the main diagonal (including the diagonal).
- It inherits from the SquareMatrix class.
- The index of element u_{ij} is $n \cdot i + j - \frac{i(i+1)}{2}$ in the vector.

$$u_{ij} = \begin{cases} \texttt{vec}\left[n \cdot i + j - \frac{i(i+1)}{2}\right] & \text{if } i \le j \\ 0 & \text{otherwise} \end{cases}$$

Methods:
Constructor initializes an upper triangular matrix with given size:

```cpp
UpperTriangularMatrix(const int size); // Already made available
```

Copy constructor creates a deep copy of another upper triangular matrix:

```cpp
UpperTriangularMatrix(UpperTriangularMatrix const& other);
```

Matrix addition, subtraction, and multiplication operators:

```cpp
UpperTriangularMatrix operator+(const UpperTriangularMatrix& other) const;
UpperTriangularMatrix operator-(const UpperTriangularMatrix& other) const;
UpperTriangularMatrix operator*(const UpperTriangularMatrix& other) const;
bool operator==(const UpperTriangularMatrix& other) const;
```

Accessors for upper triangular matrix elements:

```cpp
double getElement(const int row, const int col) const;
void setElement(const int row, const int col, const double value);
```

Problem 1 continued on next page. . .

Diagonal Matrix

Definition 5 (Diagonal Matrix). *A Diagonal Matrix D is a square matrix where all off-diagonal elements are zero:*

$$D = [d_{ij}]_{n \times n} = \begin{bmatrix} d_{0,0} & 0 & 0 & \cdots & 0 & 0 \\ 0 & d_{1,1} & 0 & \cdots & 0 & 0 \\ 0 & 0 & d_{2,2} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & d_{n-2,n-2} & 0 \\ 0 & 0 & 0 & \cdots & 0 & d_{n-1,n-1} \end{bmatrix}$$

The diagonal matrix D is represented as a one-dimensional vector of size n:

$$D = \begin{bmatrix} d_{0,0} & 0 & 0 \\ 0 & d_{1,1} & 0 \\ 0 & 0 & d_{2,2} \end{bmatrix}$$

```
std::vector<double> vec = [d[0][0], d[1][1], d[2][2]]
```

Properties:

- There are n rows and n columns.
- The total number of elements is n.
- Storage optimization by only storing diagonal elements.
- It inherits from the SquareMatrix class.
- The index of element d_{ij} is i in the vector.

$$d_{ij} = \begin{cases} \texttt{vec}[i] & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Methods:
Constructor initializes a diagonal matrix with given size:

```cpp
DiagonalMatrix(const int size);
```

Copy constructor creates a deep copy of another diagonal matrix:

```cpp
DiagonalMatrix(DiagonalMatrix const& other);
```

Matrix addition, subtraction, and multiplication operators:

```cpp
DiagonalMatrix operator+(const DiagonalMatrix& other) const;
DiagonalMatrix operator-(const DiagonalMatrix& other) const;
DiagonalMatrix operator*(const DiagonalMatrix& other) const;
bool operator==(const DiagonalMatrix& other) const;
```

Accessors for diagonal matrix elements:

```cpp
double getElement(const int row, const int col) const; // Already made available
void setElement(const int row, const int col, const double value);
```

Problem 1 continued on next page. . .

You have to implement all 5 classes given above along with the functions stated. For each class one of the functions has already been provided.

Grading Rubric:

Component	**Points**
Matrix Class Functions (11)	30
SquareMatrix Class Functions (7)	20
UpperTriangularMatrix Class Functions (7)	10
LowerTriangularMatrix Class Functions (7)	10
DiagonalMatrix Class Functions (7)	10
All Test cases passing	20
TOTAL	100