Wayne State University

Department of Computer Science

Ph.D. Proficiency Exam

Object-Oriented Programming Skills

In mathematics, a matrix (plural: matrices) is a rectangular array of numbers, symbols, or expressions, arranged in rows and columns. For example, two matrices A and B are defined as following:

$$\mathbf{A}=\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} \qquad \mathbf{B}=\begin{vmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{vmatrix}$$

Operations supported by Matrices include:

i. **Addition** with another matrix of the same dimension**:**

$$\mathbf{A + B} = \begin{vmatrix} a_{11}+b_{11} & a_{12}+b_{12} & a_{13}+b_{13} \\ a_{21}+b_{21} & a_{22}+b_{22} & a_{23}+b_{23} \\ a_{31}+b_{31} & a_{32}+b_{32} & a_{33}+b_{33} \end{vmatrix}$$

ii. **Subtraction** from another matrix of the same dimension:

$$\mathbf{A - B} = \begin{vmatrix} a_{11}-b_{11} & a_{12}-b_{12} & a_{13}-b_{13} \\ a_{21}-b_{21} & a_{22}-b_{22} & a_{23}-b_{23} \\ a_{31}-b_{31} & a_{32}-b_{32} & a_{33}-b_{33} \end{vmatrix}$$

iii. **Transpose (***a_{ij} becomes a_{ji} in the transposed Matrix):*

$$\mathbf{A}^{\mathbf{T}} \text{ (Transpose)} = \begin{vmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{vmatrix}$$

iv. **Scalar multiplication:** Multiply each element with a scalar number

$$\mathbf{2A} = \begin{vmatrix} 2a_{11} & 2a_{12} & 2a_{13} \\ 2a_{21} & 2a_{22} & 2a_{23} \\ 2a_{31} & 2a_{32} & 2a_{33} \end{vmatrix}$$

v. **Multiplication:** If **A** is an *n* × *m* matrix and **B** is an *m* × *p* matrix,

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mp} \end{pmatrix}$$

the *matrix product* **C** = **AB** (denoted without multiplication signs or dots) is defined to be the *n* × *p* matrix

$$\mathbf{C} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1p} \\ c_{21} & c_{22} & \cdots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{np} \end{pmatrix}$$

Where, $c_{ij} = a_{i1}b_{1j}+ a_{i2}b_{2j} + \ldots \ldots \ldots + a_{im}b_{mj}$ for i = 1, ..., n and j = 1, ..., p. *Please note that multiplication requires that the number of rows in the first matrix should be equal to the number of columns in the second matrix.*

Requirements

Implement a `Matrix` class with following properties:

1. Stores a matrix of integers. It must have a constructor that takes the number of rows and the number of columns as the arguments. For example, the signature of the constructor can be:

   ```
   Matrix(int nRows, int nCols);
   ```

2. Dynamically allocate memories for the Matrix in the constructor. The constructor must check invalid arguments (i.e., row or column cannot be less than 1). All the member variables must be declared as private. You should implement necessary getters and setters. Specially, `getValueAt(int r, int c)` and `setValueAt(int r, int c)` would be very useful. *If you are implementing in C++, please make sure that you're implementing a copy constructor. Otherwise, you will encounter segmentation faults when returning Matrix objects from methods. The syntax to declare a copy constructor in c++ is:* `Matrix (const Matrix &object)`

3. Implement a method to load the Matrix from an array (e.g. `bool loadFromArray(int[] items)`). The array would contain nRows X nCols integers and in a rowwise sequence. For example, to load the Matrix A defined in the page 1, your array would be: [a_{11}, a_{12}, a_{13}, a_{21}, a_{22}, a_{23}, a_{31}, a_{32}, a_{33}]

4. Implement addition operation that adds another Matrix to the current Matrix and returns the result.

   ```
   Matrix addition(Matrix B);
   ```

5. Implement subtraction operation that subtracts another Matrix from the current Matrix.

   ```
   Matrix subtraction(Matrix B);
   ```

6. Implement transpose operation that returns the transpose of the current Matrix.

   ```
   Matrix getTranspose();
   ```

7. Implement scalar multiplication operation that returns the product.

   ```
   Matrix multiplyBy(int scalar);
   ```

8. Implement multiplication operation that returns the product of the two matrices.

   ```
   Matrix multiplyBy(Matrix B);
   ```

9. Implement a method to print the matrix in a properly understandable format.

10. Implement a main method that creates following three matrices using the loadFromArray method.

$$\mathbf{X}= \begin{vmatrix} 5 & 0 & -5 \\ 2 & 4 & 2 \\ -1 & 3 & 9 \end{vmatrix} \qquad \mathbf{Y}= \begin{vmatrix} 1 & 6 & -2 \\ -1 & 8 & -5 \\ 3 & 4 & 10 \end{vmatrix} \qquad \mathbf{Z}= \begin{vmatrix} 2 & 0 \\ 1 & 5 \\ -9 & -3 \end{vmatrix}$$

Compute and print following matrices:

a) 2X+Y

b) 2Y-X

c) Z^T

d) XYZ

Instructions:

1. You can use either C++ or Java.
2. **DO NOT use the internet (no exception). Any attempt will result in an expulsion from the exam.**
3. No question is allowed during the exam. If something is unclear, write down your assumptions as comments in the code or in the README.txt file.
4. Please make sure that your program checks all the error conditions/ exceptions (e.g., if dimensions mismatch during matrix addition/subtraction) and displays appropriate messages.
5. Store all the source files and the executable file on the diskette. You should also provide detail information, such as software environment used, instructions to build, execute, and test your program. Please make sure to write down the version of the C++ or Java compiler that you used in the README.txt file.
6. If your program does not compile, you will be given a zero grade. Therefore, please ensure that it compiles correctly before submission.
7. Make sure that your identifier names are meaningful, and code is properly indented.
8. Time limit is 3 hours. Late submissions will not be accepted.
9. Do not use any comments or identifiers in your files that may reveal your identity.