COMP 2012H: Honors OOP and Data Structures
Fall 2015

PA3: Matrix ADT
Deadline: 11:59pm, Sunday, October 25, 2015

---

1 Matrix ADT

In this assignment, you are asked to implement a `Matrix` ADT. The ADT is written in a .h file, while the functions are fully implementated in a .cpp file.

The index of our matrix starts from 0, and hence an $m \times n$ matrix $A_{m\times n}$ is written as $(m, n \geq 0)$

$$A_{m\times n} = \begin{pmatrix} a_{00} & a_{01} & \cdots & a_{0,n-1} \\ a_{10} & a_{11} & \cdots & a_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m-1,0} & a_{m-1,1} & \cdots & a_{m-1,n-1} \end{pmatrix},$$

where $a_{i,j}$ are `double` numbers, and $m, n \geq 0$.

The `Matrix` class definition is as follows. You should implement your matrix as a dynamic 2-D array and provide codes for all its member functions. You MUST keep all the shown functions, but may add your private data members and/or member functions if you want.

```
class Matrix {                              // Matrix class
public:
  explicit Matrix(int rows = 0, int cols = 0);
                                            // constructor
  Matrix( const Matrix & mt);               // copy constructor
  ~Matrix();                                // destructor
  int rows() const;       // return the number of rows of the matrix
  int cols() const;       // return the number of columns of the matrix
  double & el(int i, int j) const;            // access (i, j)th element
  void assign(const Matrix & op);           // copy values from op
  Matrix mul(const Matrix & op) const;  // multiplication of matrices
  Matrix transpose() const;                 // the transpose of matrix
  Matrix inverse() const;               // the inverse of matrix

// Add your public member functions, if any, in the following
private:
  double **elm;                                // matrix elements
  int r;                                    // number of rows
  int c;                                    // number of columns
// Add your private data members and private member functions,
// if any, in the following
};
```

The member functions are explained as follows.

1. Constructor:

```
// Constructor: Initialize the matrix to be a rows x cols matrix.
// rows >= 0; cols >= 0
// default is an empty 0 x 0 matrix
// No need to initialize matrix elements
Matrix::Matrix(int rows, int cols) {
```

2. Copy constructor:

```
// Copy constructor
Matrix::Matrix(const Matrix & mt) {
```

3. Destructor.

```
// Destructor for the matrix
Matrix::~Matrix() {
```

4. Inspector function for row:

```
// Return the number of rows of the matrix
int Matrix::rows() const {
```

5. Inspector function for column:

```
// Return the number of columns of the matrix
int Matrix::cols() const {
```

6. Element access:

```
// Return the (i, j)th element of the matrix
// Precondition: i and j are valid ranges
double & Matrix::el(int i, int j) const {
```

7. Assignment function, to copy all elements in matrix `op`. That is, a call of `A.assign(B)` will copy element by element from matrix `B` to `A`.

   Note that you need to resize the matrix before copying so that the new matrix is of the same dimension as `op`.

```
// Assign and copy all the elements of matrix op to the matrix.
// Resize matrix (allocate space) if necessary
void Matrix::assign(const Matrix & op) {
```

8. Matrix multiplication, where a call of `A.mul(B)` returns a new matrix which is the product of matrices `A` and `B`.

   Matrix multiplication is defined as follows. The product of two matrices $\mathtt{A}_{m\times p}$ (with entries a_{ij}) and $\mathtt{B}_{p\times n}$ (with entries b_{ij}) is a $C_{m\times n}$ matrix whose entries c_{ij} is given by

$$c_{ij} = \sum_{k=0}^{p-1} a_{ik} b_{kj}, \quad \forall 0 \le i < m,\ 0 \le j < n.$$

```
// Return a new matrix which is the product of
// this matrix and matrix op.
// Precondition: valid multiplication with correct rows and columns
Matrix Matrix::mul(const Matrix & op) const {
```

9. Transpose operation, which returns a new matrix which is the transpose of this matrix.

   Matrix $B_{n\times m}$ is the transpose of matrix $A_{m\times n}$ iff

$$b_{ji} = a_{ij}, \quad \forall 0 \le i < m,\ 0 \le j < n.$$

```
// Return a new matrix which is the transpose of the matrix.
Matrix Matrix::transpose() const {
```

10. Inverse: Matrix B is the inverse of a square matrix A iff $AB = I$. If A does not have an inverse, please return a zero matrix. (For a discussion on how to inverse a matrix, you may consult `http://www.mathwords.com/i/inverse_of_a_matrix.htm`) Hint: The Adjoint method described in the webpage is easier to implement than the others.

```
// Return a new matrix which is the inverse of the matrix.
// Return a zero matrix if inverse does not exist
Matrix Matrix::inverse() const {
```

2 What to be turned in

Provide us your header file `Matrix.h` and the implementation file `Matrix.cpp`. We will include your `Matrix.h` and write our own tester to test-run your program. However, if you have a tester file, please provide to us as well.

3 Extra Credits (Maximum 10%)

There are many other matrix operations, such as eigenvectors, eigen-values, solution of $Ax = b$, etc. Implement what you know about matrix operations to gain extra credits here (at the discretion of the grader). Please explain the extras you have done in your README.