HW #5 Triangular Matrix

- In this assignment you are to design, implement, and test a class called **TriangularMatrix** using a one-dimensional dynamic array.
- The program must be written in C++.

- A triangular matrix is a matrix where the nonzero elements are confined in one half of the matrix and the other half contains only zero.

HW #5 (2)

- For this assignment we only consider matrices for which the number of columns is same as the number of rows.
- We can define two types of triangular matrices, e.g., lower triangular and upper triangular as shown in the figure where *X* denotes an element which can be a zero or a nonzero.

$$\begin{pmatrix} X & 0 & 0 & 0 \\ X & X & 0 & 0 \\ X & X & X & 0 \\ X & X & X & X \end{pmatrix} \quad \begin{pmatrix} X & X & X & X \\ 0 & X & X & X \\ 0 & 0 & X & X \\ 0 & 0 & 0 & X \end{pmatrix}$$

HW #5 (3)

- The matrix on the left is lower triangular and the matrix on the right is upper triangular.
- To implement a triangular matrix we must utilize this special information, i.e., we need not use memory to store matrix elements that are knows to be zero. In other words, we need a dynamic array of size

$$\sum_{i=1}^{n} i = n(n+1)/2$$

HW #5 (4)

to store only the elements that are marked *X* for an *n* x *n* triangular matrix. This way we save almost one half the space that is needed for a full matrix of dimension *n* x *n*.

- Implement the following member functions of the class and write the implementation in the file **TriangularMatrix.cc**.

```
Matrix (int n=2); // create an n x n matrix
Matrix (const Matrix& m); // copy constructor
~Matrix(); // destructor
Matrix& operator=(const Matrix& m); // copy assignment
```

HW #5 (5)

```
int size() const; // return the size of the matrix
Matrix& operator+=(const Matrix& m);
Matrix& operator-=(const Matrix& m);
Matrix& operator*=(const Matrix& m);

Matrix& operator*=(double s); // scalar multiplication-assignment

double operator()(int i, int j); const; // r-value
double& operator()(int i, int j); // l-value
```

HW #5 (6)

```
Matrix operator+(const Matrix& lt, const Matrix& rt);
Matrix operator-(const Matrix& lt, const Matrix& rt);
Matrix operator*(const Matrix& lt, const Matrix& rt);

Matrix operator*(const Matrix& lt, double s); // scalar multiplication
Matrix operator*(double s, const Matrix& rt); // scalar multiplication
                                              // (commutative)

ostream& operator<<(ostream& out, const Matrix& x);
void readMatrix(); // member function to read a matrix from users
```

HW #5 (7)

- Write a test program (in **testMatrix.cc**) showing the use of each member function that you have implemented.
- In your test program you should use **readMatrix()** (a member function) to input a matrix from the user interactively (i.e., number of rows, columns, whether lower or upper triangular, and the elements of the matrix). No file input/output is required.