CSE221 Data Structures (Fall 2015)
Instructor: Prof. Won-Ki Jeong
Due date: Oct 11, 2015, 11:59 pm.

Assignment 3: Sparse Matrix Representation using Linked Lists

In this assignment, you will implement a sparse matrix data structure using linked lists, as shown in the textbook (p.219-222). You are required to modify/complete `matrix.txx` only.

1. Matrix class (100 pts)

In matrix.h, the matrix abstract data type is defined. You will find function names as well as data. Actual implementation of those functions should be in matrix.txx. You need to implement following functions:

- Matrix(int row, int col) : constructor that creates an empty matrix of size row x col.
- Matrix(std::ifstrea& is) : constructor that creates a matrix from a text file
- Matrix(const Matrix& a) : constructor that creates a matrix from another matrix
- ~Matrix() : destructor
- Matrix& operator=(const Matrix& a) : assignment operator
- Matrix operator+(const Matrix& b) : Add current matrix and b
- Matrix operator-(const Matrix& b) : Subtract current matrix and b
- Matrix operator*(const Matrix& b) : Multiplication current matrix and b
- Matrix& SwapRow(int i, int j) : Swap i-th and j-th rows
- Matrix& SwapCol(int i, int j) : Swap i-th and j-th column
- Matrix& CreateElement(int i, int j, type val) : Create new element into current matrix. If (i,j) is out-of-range, then discard it. If there already exists an element for (i,j), then replace the current value with the new one.
- Matrix& Transpose() : Transpose matrix

For +, -, and * operators, if the matrix dimensions are not correct (for example, size of two matrices should be same for + and – operators), you must return an empty matrix and print out the error message "DIMENSION MISMATCH".

Note that +, -, * operators are right-hand side operators (meaning that A + B

then + operator from A will be called and B will be the input operand). In addition, note that functions return Matrix does not destroy the current matrix, but functions return Matrix& will destroy (modify) the current matrix.

For example,

A = B.Transpose();

This will first transpose B, and copy it to A, so A and B will be the same matrix after this line.

However,

A = B + C;

After this, B and C will not be changed. That means, + operator will create a new matrix that is a sum of B and C. Then the assignment operator = will copy the value of that matrix to A.

You need to read the textbook p.216~222. The matrix class for this assignment is similar to the one given in the textbook except the followings:

1. Matrix class you need to implement is a template class. That means, value type can be defined by the users (in the textbook, only integer value can be used).
2. The header node of the header list is same as a regular header node, and it does not store # rows / # cols / # elements. Instead, Matrix class has nRow, nCol and nElem variables to store such values.

You also need to make sure the followings:

1. Row / Column index starts with 0.
2. Input matrix.txt is ordered by rows and within rows by columns.
3. If you create a new element, it must be inserted into the correct location (row and column lists must be in ascending order)

2. Compile and submit

You must log in uni06~10.unist.ac.kr for coding and submitting the assignment.

You can compile the code using the included Makefile. You can simply `make` and then the code will be compiled. The output executable name is `assign_3`.

You must write a report describing your work in this assignment. If you are ready to submit the code and the report, then make a zip file containing the files (`matrix.txx` and the report file) and use the `dssubmit` script as follows:

```
> dssubmit assign3 assign3.zip
```