CMP 210 – Data Structures and Algorithms
Fall 2019
LAB-5

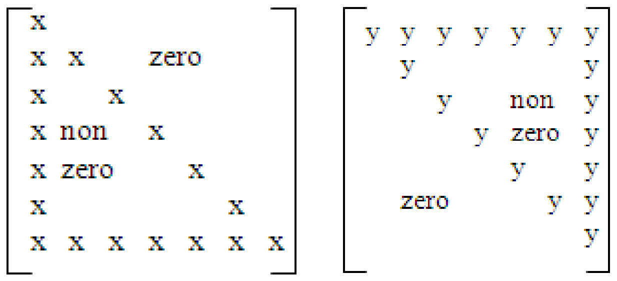
Issue Date: Oct 24, 2019

ALERT!

1. The objective of this lab is understanding sparse matrices and addressing formula.
2. This is an individual lab, you are strictly **NOT** allowed to discuss your solution with fellow colleagues, even not allowed asking how is he/she is doing, it may result in negative marking. You can **ONLY** discuss with your TAs or with me.
3. Beware of memory leaks and dangling pointers.
4. Pay attention to **GOOD coding conventions** e.g.
   - Proper indentation.
   - Meaning variable and function names.
   - Use camelCase naming convention
   - Use meaningful prompt lines/labels for all input/output
5. **Anyone caught in act of plagiarism would be awarded an "F" grade in this Lab.**

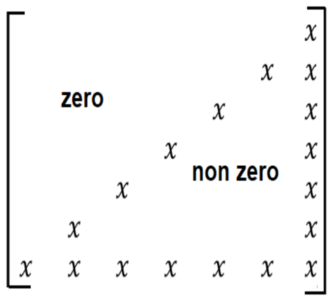
Task 01: [5+5 Marks]

When all elements either above or below the main diagonal of a square matrix are zero, then matrix is said to be triangular. Figure below shows a lower and an upper triangular matrix.

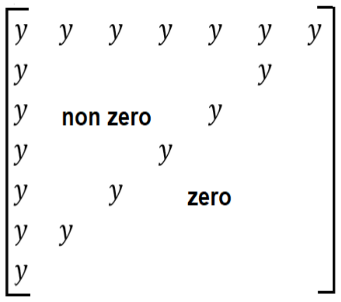


We have already discussed the above two (lower left and upper right) triangular matrices and their addressing formulae if the matrix is stored in a linear array in row major order. Now obtain an addressing formula for elements in the:

a) Lower right triangle if it is stored in row major order in a linear array with A[0][0] being the first element.



b) Upper left triangle if it is stored in row major order in a linear array with A[0][0] being the first element.



Task 02: [10 Marks]

Develop a class for 2D matrices of any generic type (using templates), This class should store the elements of the 2-D matrix in a linear array of generic type created dynamically. Thus you will have to use a mapping formula to store and retrieve items.

Your class should support following operations:

1. **Constructor, destructor, Copy-constructor.**
   You should always implement constructor, destructor and copy-constructor in case of dynamic memory allocation.
2. **getElement (i, j)**
   Get the value of element stored at i^{th} row and j^{th} column.
3. **setElement(i, j, val)**
   Set the value of element stored at i^{th} row and j^{th} column.
4. **printMatrix( )**
   This function should print the matrix on console (in 2D matrix form).
5. **printMatrix(matrix)**
   This function should print the given matrix on console (in 2D matrix form).
6. **transpose ( )**
   This function should take the transpose of the matrix.
7. **printSubMatrix(r1, r2, c1, c2)**
   This function should display the sub matrix specified by given arguments.
8. **Overload + operator**
   To adds two matrices.
9. **clear(n)**
   To clear the first n rows and columns of the matrix.

Write a main program providing menu to make it easy to use and test matrix class functionalities. No marks shall be given without this driver program.

Task 03: [5+5 Marks]

Write a function which prints the Row-Major based ND-array formula against a given number of dimensions. The header of the function is given bellow

```
void printND(int dimensions)
```

For example if the function is called for 3 dimensions i.e printND(3) then it should print

$$i_1D_2D_3+i_2D_3+i_3$$

Here, i_1, i_2,i_3 represent the index set and D_1, D_2,D_3 represents dimension set. Also write a main program to test this function with appropriate messages.