CSCI-SHU 210 Data Structures

Homework Assignment 4 Dynamic Arrays (2-D)

There are several approaches that we can use to store and organize the data for a 2-D array. Two of the more common approaches include the use of a single 1-D array to physically store the elements of the 2-D array by arranging them in order based on either row or column, whereas the other uses an array of arrays. We are going to use the latter approach to implement the 2-D Array abstract data type.

In the Python file, you will be given two classes: class UserDefinedDynamicArray and class UDDA_Matrix.

- class UserDefinedDynamicArray is the class we defined in the lecture, but with very basic methods allowed only. **You should not modify** class UserDefinedDynamicArray.

- class UDDA_Matrix is built on class class UserDefinedDynamicArray. Some basic methods has been written for you. You should not modify them, **but you are suggested to take a look** to understand the implementation.

```
Class UDDA_Matrix:
__init__( self, numRows, numCols )       # Constructor
__getitem__( self, ndxTuple )            # Support Matrix[row, column], Matrix[row]
get_column(self, col)                    # Support Matrix[col]
__setitem__( self, key, value )          # Support Matrix[row, column] = value, Matrix[row] = value
set_column(self, col, value)             # Support Matrix[col] = value
max_value(self)                # Return the max value within the n x n matrix      #Task 1
scaleBy( self, scalar )                  # Support Matrix * integer                 #Task 2
tranpose( self )                                                                    # Task 3
__add__( self, rhsMatrix )               # Support Matrix1 + Matrix2                 #Task 4
__sub__( self, rhsMatrix )               # Support Matrix1 − Matrix2                 # Task 5
__mul__( self, rhsMatrix )               # Support Matrix1 * Matrix2                 # Task 6
__str__(self)                            # Support printing
```

Guidelines & hints:
1. Only modify **six** functions (Task 1, 2, 3, 4, 5, 6) to finish this homework.
2. Please make sure the test code provided runs without any modification to the test code.
3. You don't have to handle matrix size error. I will test your implementation with only valid matrix sizes.
4. To access row i, column j, enter matrix[i, j], or self[i, j] if you are coding within matrix class.
5. Your implementation should also work for larger matrixes.

Task 1: max_value(self)

Return the maximum value of this matrix. This matrix should remain unchanged after the function call.

And in our case,

```
>>> m1 = UDDA_Matrix(3,2)        # 3 Rows, 2 Columns

######### add some values in matrix m1 with code

>>> print(m1.max()
```
If m1 is
```
9            6
7            8
6            0

Then
9


Should be printed.
```

Task 2: scaleBy(self, scalar)

Scales the original matrix by the given scalar.
Return: nothing. **Modification is done on original matrix.**

We have scaling operations as follows. Here we scale the matrix by a factor of 3.

$$3 \begin{bmatrix} 6 & 7 \\ 8 & 9 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 3*6 & 3*7 \\ 3*8 & 3*9 \\ 3*1 & 3*0 \end{bmatrix} = \begin{bmatrix} 18 & 21 \\ 24 & 27 \\ 3 & 0 \end{bmatrix}$$

And in our case,

```
>>> m1 = UDDA_Matrix (3,2)
### add some values in the matrix with code
>>> m1.scaleBy(3)
>>> print(m1)
if m1 is
9          6
7          8
6          0

Then
27         18
21         24
18         0

Should be printed.
```

Task 3: transpose(self)

Creates and **returns a new matrix** that is the transpose of this matrix.
The original matrix should not be modified.

$$\begin{pmatrix} 5 & 4 \\ 4 & 0 \\ 7 & 10 \\ -1 & 8 \end{pmatrix}^{T}_{4 \times 2} = \begin{pmatrix} 5 & 4 & 7 & -1 \\ 4 & 0 & 10 & 8 \end{pmatrix}_{2 \times 4}$$

And in our case,

```
>>> m1 = UDDA_Matrix(3,2)
### add some values in the matrix with code
>>> print(m1.transpose())
if m1 is
9          6
7          8
6          0

Then
9          7          6
6          8          0

Should be printed.
```

Task 4:   add   (self, rhsMatrix)

Creates and **returns a new matrix** that results from matrix addition.
m1, m2 remains unchanged.

$$3+4=7$$

$$\begin{bmatrix} 3 & 8 \\ 4 & 6 \end{bmatrix} + \begin{bmatrix} 4 & 0 \\ 1 & -9 \end{bmatrix} = \begin{bmatrix} 7 & 8 \\ 5 & -3 \end{bmatrix}$$

And in our case,

```
>>> m1 = UDDA_Matrix(3,4)
>>> m2 = UDDA_Matrix(3,4)
### Some Code to fill matrix
>>> print(m1 + m2)If
m1 is
9          6          -1         -6
6          8          -4         7
-3         -6         4          9
If m2 is
-8         9          -1         -4
-6         -1         4          -4
-2         6          7          0
Then
1          15         -2         -10
0          7          0          3
-5         0          11         9
Should be printed.
```

Task 5:   sub   (self, rhsMatrix)

Creates and **returns a new matrix** that results from matrix subtraction.
m1, m2 remains unchanged.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} - \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1-9 & 2-8 & 3-7 \\ 4-6 & 5-5 & 6-4 \\ 7-3 & 8-2 & 9-1 \end{bmatrix}$$

$$= \begin{bmatrix} -8 & -6 & -4 \\ -2 & 0 & 2 \\ 4 & 6 & 8 \end{bmatrix}$$

And in our case,

```
>>> m1 = UDDA_Matrix(3,2)
>>> m2 = UDDA_Matrix(3,2)
### Some Code to fill matrix
>>> print(m1 − m2)If
m1 is
5          2
4          3
7          9
If m2 is
4          3
1          1
1          6
Then

1          -1
3          2
6          3
Should be printed.
```

Task 6:   mul   (self, rhsMatrix)

Creates and **returns a new matrix** resulting from matrix multiplication.

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix}, B = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{pmatrix}$$

$$AB = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} & a_{11}b_{13} + a_{12}b_{23} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} & a_{21}b_{13} + a_{22}b_{23} \\ a_{31}b_{11} + a_{32}b_{21} & a_{31}b_{12} + a_{32}b_{22} & a_{31}b_{13} + a_{32}b_{23} \end{pmatrix}$$

$$BA = \begin{pmatrix} b_{11}a_{11} + b_{12}a_{21} + b_{13}a_{31} & b_{11}a_{12} + b_{12}a_{22} + b_{13}a_{32} \\ b_{21}a_{11} + b_{22}a_{21} + b_{23}a_{31} & b_{21}a_{12} + b_{22}a_{22} + b_{23}a_{32} \end{pmatrix}$$

And in our case,

```
>>> m1 = UDDA_Matrix(3,2)
>>> m2 = UDDA_Matrix(2,3)
### Some Code to fill matrix
>>> print(m1 * m2)
If matrix 1 is:
5          4
4          5
1          8

matrix 2 is:
8          0          6
2          0          1

Then

48         0          34
42         0          29
24         0          14
Should be printed.
```