

## 1. (60 points) Matrix Class

A matrix is rectangular array of items laid out in rows and columns. The dimensions, or size, of a matrix can be expressed as  $m \times n$  or  $m$ -by- $n$ , where  $m$  is the number of rows in the matrix and  $n$  is the number of columns in the matrix.

For example, consider  $A$ , which is the following  $2 \times 4$  matrix:

$$\begin{bmatrix} 5 & 1 & 2 & 3 \\ 3 & 4 & 4 & 1 \end{bmatrix}$$

The individual elements in  $A$  can be expressed as  $a_{ij}$ , where  $i$  (the row) is a number from 1 to  $m$  and  $j$  (the column) is a number from 1 to  $n$ . For example, the value at element  $a_{1,3}$  is 2.

Write a program (called *matrix.cpp*) that does that following:

### 1. Implement a class called *Matrix* that:

- Contains private member fields for the number rows and columns of the matrix
- Contains a public member field to contain the matrix elements
  - This should be a 2D array of integers that is implemented dynamically
- Contains five public functions
  - add function, that adds two same sized matrices together and returns a new matrix with the result
  - subtract function, that subtracts two same sized matrices together and returns a new matrix with the result
  - multiply function, that performs proper matrix multiplication and returns a new matrix with the result
  - scalar function, that performs scalar multiplication with an integer value and a matrix, and returns a new matrix with the result
  - print function, that outputs the contents of the matrix in tabular form that matches the dimensions of the matrix
- Contains a non-default constructor
  - Constructor that accepts size information, and dynamically creates the matrix
- Contains a destructor
  - That properly handles discarding the dynamically created 2D array (using delete and setting the member field to null)

### 2. Prompts the user for:

- The dimensions of a first matrix
- The contents of the first matrix, which is then filled into the newly created matrix object instance.
- The dimensions of a second matrix
- The contents of the second matrix, which is used to fill the newly created matrix instance

Sample prompts with appropriate user responses:

Number of Rows in Matrix 1: 5  
Number of Columns in Matrix 1: 2  
Values of Matrix 1 (expecting 10): 6 7 10 3 5 31 0 9 2

**Note: You must use the above format for entering the values of the matrix. When entering values to fill a matrix, all values should be provided on one line.**

3. Performs the following calculations and prints each result using the *print* function
  - Each of the four matrix mathematical methods should be called, each result stored in a new object
  - If the dimensions of the two matrices involved do not allow for the operation to be performed, skip performing this calculation, and display a message stating that step has been skipped.
    - For example, if I have a 3x4 matrix and a 4x2 matrix, I cannot add or subtract these together, but I can perform multiplication
  - Each calculation should be printed with a full explanation
    - For scalar multiplication, you can either use an integer literal or generate a random integer. Just be sure to print the value of the integer value as part of the output when performing this function.
    - The calculation being performed should be explained; and the contents of each matrix or value involved should be printed and identified.

## 2. (40 points) Matrix Class with Overloaded Operators

Using the class from the first portion of the assignment, write a program (called *matrix\_ops.cpp*) that does the following:

1. Overloads the following operators using *friend*:
  - +  
Replicate the functionality of the add function from *matrix.cpp*
  - -  
Replicate the functionality of the subtraction function from *matrix.cpp*
  - \*  
Overload this to replicate the functionality of both the matrix multiplication and scalar multiplication functions from *matrix.cpp*  
  
This operator should be overloaded a total of 3 times.
  - <<  
Replicate the functionality of the print function from *matrix.cpp*  
Used in the following way: *cout << matrix\_instance1;*
  - >>  
Used as the only way to fill in a matrix with values  
Used in the following way: *cin >> matrix\_instance1;*
2. Remove the five public functions of the Matrix class, and use only overloaded operators to perform the same functionality/output as in the *matrix.cpp*

3. Performs the following calculations and prints each result using the overloaded >> operator
  - Each of the four matrix mathematical methods should be called, each result stored in a new object. Be sure you invoke scalar multiplication once for each overloaded operator
    - `int * Matrix`
    - `Matrix * int`
  - If the dimensions of the two matrices involved do not allow for the operation to be performed, skip performing this calculation, and display a message stating that step has been skipped.
    - For example, if I have a 3x4 matrix and a 4x2 matrix, I cannot add or subtract these together, but I can perform multiplication
  - Each calculation should be printed with a full explanation
    - For scalar multiplication, you can either use an integer literal or generate a random integer. Just be sure to print the value of the integer value as part of the output when performing this function.
    - The calculation being performed should be explained; the contents of each matrix or value involved should be printed and identified.

## Compiling the Program

Use the following command to compile your classes:

```
g++ -Wall -o <output_name> <program_name.cpp>
```

Example:

```
g++ -Wall -o matrix matrix.cpp
```

**Remember:** Your code must successfully compile without any warnings or errors, or a zero will be given for the assignment.

## Submission

- Electronic Submission (Due: One minute before midnight, 11:59 PM, February 7, 2018)
  - Your two source code files (*matrix.cpp*, *matrix\_ops.cpp*)
  - Submission instructions will be provided during lecture and then sent out via email
- Hardcopy Submission (Due: Beginning of class, February 8, 2018)
  - Printed hardcopies of each of the two source code files
  - The pages of each program should be stapled together
  - Submitted in a pocket folder with your name, the course, and the section number on the front