CMPSC 122.2
February 1, 2022

Homework 1

Due: February 15, 2022

---

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 x 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 $\boldsymbol{A}$, which is the following 2 x 4 matrix:

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

The individual elements in *A* can be expressed as $a_{i,j}$, 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 20, 2020)

  - Your two source code files (*matrix.cpp*, *matrix_ops.cpp*)
  - Zip file of source code (can include optional README file to demonstrate how to run your program) submitted via CANVAS drop box by close of assignment