CSCE 221 Cover Page
Homework Assignment # 1

First Name: Kirsten Last Name: Madina UIN: 626003641

User Name: kirsten.madina E-mail address: kirsten.madina@gmail.com

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more in the Aggie Honor System Office http://aggiehonor.tamu.edu/

Type of sources					
People	Austin Peterson				
Web pages (provide URL)	Stack Overflow	Piazza Forums	CPlusPlus		
Printed material	n/a				
Other Sources	n/a				

I certify that I have listed all the sources that I used to develop the solutions/code to the submitted work.

"*On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work.*"

Your Name (signature) Kirsten Madina Date 2/4/2019 The Programming Assignment Report

Part I
Assignment Description

In this programming assignment, we are given skeleton code for the class My_matrix, which is a two dimensional array of the type int. Our task is to implement all the functions declared in the .h file and to program 5 different ways to test the scope of the class. Our second task is to create a generic or Templated version of My_matrix that can hold a two dimensional array of any data type.

Part II
Data Structures and Algorithms

(a) The ADT used is a 2-dimensional array.
(b) It is implemented by creating a pointer to a pointer of the specified data type (for the first task, the data

type would be int while for the second task it would be generic numerical type T), the inner/nested pointer then pointing to an array of user-specified or defaulted 'm' elements and the outer pointer then pointing to a user-specified or defaulted 'n' arrays.
(c) There were a few algorithms used to implement the class My_matrix and perform operations. To overload the operator+, the algorithm created a new My_matrix object, checked to see if the two matrices being added were compatible, and then for each element of matrix 1 at position (i,j) to the element of matrix 2 at position (i,j) and returned it to the new matrix object. To overload the operator*, the algorithm created a new My_matrix object, checked to see if the two matrices were compatible by comparing the number of columns in matrix 1 to the number of rows in matrix 2, and then for every row in matrix 1 (i) and column in matrix 2(j) and column in matrix 1 (k), we summed the element in matrix 1 at position (i,k) mulitplied by the element in matrix 2 at position (k,j).
(d) The algorithm for operator + is n^2+1 or $O(n^2)$ while the algorithm for operator* should be n^3+1 or $O(n^3)$.

Part III
C++ Organization

(a) The classes in this programming assignment ar My_matrix and TemplatedMy_matrix. Both these classes have private int member variables n and m corresponding to row and column size. They have a private member function "void allocate_memory". They also have a pointer to a pointer ptr (of type int for My_matrix and type T for TemplatedMy_matrix). Public interfaces of this class include: the operator() which accesses an element at user-specified position (i,j) without checking bounds; function elem(int n, int m) which accesses an element at position (n,m) while checking bounds; functions number_of_rows and number_of_columns which return the n and m value of the object respectively; and operator+ and operator* can be used on objects of this class type.
(b) in the header file, "My_matix.h" and "TemplatedMy_matrix.h", the were the constructors, assignment operators, number of rows() and number of columns(), operator() overload, elem(int,int), operators >> and << overload, and operators * and + overload.
(c) The second portion of the programming assignment involves creating a Templated version of the first portion of the assignment.

Part IV
User guide

(a) Program is in directories accessed by CS221\madina_3641_PA1\Phase_1 and CS221\madina_3641_PA1\Phase_2. Files are My_matrix.h, My_matrix.cpp and main.cpp in Phase_1 and TemplatedMy_matrix.h, TemplatedMy_matrix.cpp and main.cpp in Phase_2. Both directories have a Makefile.
(b) executable is name main (./main)

Part V
Specifications and descriptions of output and input formats and files

(a) in the first phase, there is an input textfile containing a matrix titled "input.txt" that is used in Test 2, and an output textfile where the matrix is saved titled "output.txt".

(b) Input file has the following format: $\begin{matrix} n & m & \\ x & x & x \\ x & x & x \end{matrix}$ where n and m correspond to the row and column size respectively and 'x' corresponds to any numerical value within the matrix.
(c) the program could crash if 'n' and 'm' aren't specified or if the number of elements in the matrix do not correspond to the given 'n' and 'm' values.

Part VI
Exceptions

There are 2 exceptions: Invalid input and Incompatible matrices. Invalid input catches when the iput textfile isn't in the correct format and Incompatible matrices catches when two matrices whose dimensions make it so that the operation being attempted (addition or multiplication) cannot be performed.

Part VII
Testing The Program

Testing the Templated functions: I created a matrix of type int and populated it using function elem(int,int). I tested the addition operator like so:

```
TemplatedMy_matrix<int> m1(2, 2);
m1.elem(0, 0) = 1;
m1.elem(0, 1) = 2;
m1.elem(1, 0) = 3;
m1.elem(1, 1) = 4;
cout << "M1 of int: \n";
cout << m1 << endl;
```

```
cout << "M1 + M1 = \n";
cout << m1 + m1;
```

. I also created two matrices of double type and populated them the same way. I tested the multiplication operator like so:

```
TemplatedMy_matrix<double> m2(2, 2);
m2.elem(0, 0) = 1.0;
m2.elem(0, 1) = 2.0;
m2.elem(1, 0) = 3.0;
m2.elem(1, 1) = 4.0;
cout << "M2 of double: \n";
cout << m2 << endl;
```

```
TemplatedMy_matrix<double> m5(2, 3);
m5.elem(0, 0) = 1.0;
m5.elem(0, 1) = 2.0;
m5.elem(0, 2) = 3.0;
m5.elem(1, 0) = 4.0;
m5.elem(1, 1) = 5.0;
m5.elem(1, 2) = 6.0;
cout << "M5 of double: \n";
cout << m5 << endl;
```

. The output of both these operations were as expected.

```
M1 + M1 =
2 4
6 8
M2 * M5 =
9 12 15
19 26 33
```

.

Testing the int functions were similar. I populated a matrix using an input file and displayed the output like so:

```
Input file name:
input.txt
Output file name:
output.txt
Outputting m2 from input file:
1 2 3
4 5 6
7 8 9
```

I populated matrices using the () operator and attempted adding and multiplying both compatible and incompatible matrices and the outcome was as expected.

```
Multiplying m1 * m1:
7 10
15 22

Multplying m1 + m2:
Error: Incompatible matrices
Matrix m1: 1 2
3 4

Matrix m2: 1 2
3 4
5 6

Adding m1 + m1:
2 4
6 8

Adding m1 + m2:
Error: Incompatible matrices
```