Assignment 6 - Matrix Addition

Due Mar 28 by 11:59pm **Points** 20 **Submitting** a file upload **File Types** h and cpp **Available** until Mar 31 at 12:01am

This assignment was locked Mar 31 at 12:01am.

ST. CHARLES COMMUNITY COLLEGE

CPT-182 - Programming in C++

Programming Assignment - Matrix Addition (20 Points)

(Number in Question Bank: Assignment **6.1**)

Program Overview

In this assignment, you are going to write a **C++ program** that adds two matrices of the same dimensions together. Your program reads the two input matrices from an input file, and writes the matrix of the addition result to an output file.

In this assignment, you need to write a `Matrix` class, and overload the arithmetic addition operator (`'+'`) and stream insertion operator (`"<<"`) for the class.

Matrix Addition

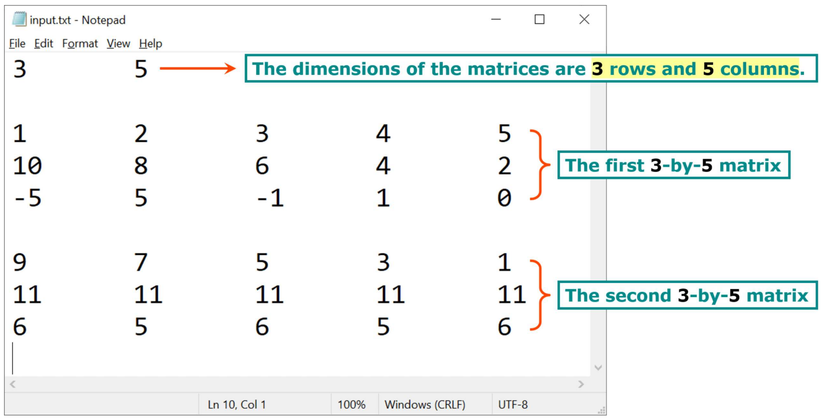
- Only the two matrices that have the **same dimensions** (same number of rows and same number of columns) can be added together.
- The addition result is **another matrix** of the same dimensions.
- Matrix addition is the procedure to add every pair of corresponding values (at the same position) together.

For example: $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} + \begin{pmatrix} 18 & 16 & 14 \\ -3 & 9 & -6 \end{pmatrix} = \begin{pmatrix} 19 & 18 & 17 \\ 1 & 14 & 0 \end{pmatrix}$

In this example, both matrices, $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ and $\begin{pmatrix} 18 & 16 & 14 \\ -3 & 9 & -6 \end{pmatrix}$, have **2** rows and **3** columns (same dimensions). The addition result is another **2**-by-**3** matrix. The value at row i column j in the result matrix is the sum of the values at row i column j in the two matrices being added together, where $0 \leq i \leq 1$ and $0 \leq j \leq 2$ in this example.

The Input File

- The input file is a **plain text file** (filename `matrices.txt`).
- First row of the input file contains **2** positive integers, which are the number of rows and number of columns (in that order) of the matrices to be added together.
- The rest of the input file lists all the element values in the two matrices to be added together. Please see the picture below to better understand the input file format.



```
3       5        The dimensions of the matrices are 3 rows and 5 columns.

1       2       3       4       5
10      8       6       4       2      The first 3-by-5 matrix
-5      5       -1      1       0

9       7       5       3       1
11      11      11      11      11     The second 3-by-5 matrix
6       5       6       5       6
```

- All the element values in the input matrices are integers.

- You **cannot** assume (or guess) the dimensions of the input matrices. In other words, no matter how large the input matrices are, your program should correctly process them.
- Please refer to the **sample input files** to better understand the input file format.

The Output File

- The output file is a **plain text file** (filename **`result.txt`**).
- First row of the output file should be written with **2** positive integers, which are the number of rows and number of columns (in that order) of the result matrix.
- Then, all the element values in the result matrix are written to the file. All values should be nicely aligned by matrix rows and columns.
- Please refer to the **sample output files** to better understand the output file format.

The `Matrix` Class

- The class contains the following data fields:
  1) **`num_of_rows`**. This is an unsigned integer variable representing the number of rows of the matrix.
  2) **`num_of_columns`**. This is an unsigned integer variable representing the number of columns of the matrix.
  3) **`data`**. This is a **2**-dimensional vector variable (type **`vector<vector<int>>`**) that stores the values in the matrix.
- The class has the following constructors:
  1) A **default constructor** that initializes both **`num_of_rows`** and **`num_of_columns`** to **0**. Here, you do **not** need to worry about the initial value of **`data`**.
  2) **Constructor** that takes **2** unsigned integers (representing the initial number of rows and number of columns) as arguments. The constructor loads **`num_of_rows`** and **`num_of_columns`** with the values passed in, and initializes **`data`** using those dimensions.
  3) The above two constructors can be combined into one constructor using **default parameter values**.
- The class contains the **getters** for data fields **`num_of_rows`** and **`num_of_columns`**. ~~Setters~~ for these two data fields are **not** required in this class.
- The class has the following class-member functions:
  1) **`get(i, j)`**. The function takes **2** unsigned integers (**`i`**: row index, **`j`**: column index) as arguments. The function returns the element value stored in the matrix at row index **`i`** and column index **`j`**.
  2) **`set(i, j, val)`**. The function takes **2** unsigned integers (**`i`**: row index, **`j`**: column index) and another integer (**`val`**: updated value of element) as arguments. The function updates the element value at row index **`i`** and column index **`j`** with the updated value passed in. The function does **not** return a value.
- The class should overload the following operators:
  1) **Arithmetic addition operator** (`'+'`). Two matrices of the same dimensions can add together to form the result matrix. Addition logic has been discussed in the **Matrix Addition** section of this assignment. It is your responsibility to correctly determine the function return type, function argument(s), whether the argument(s) should be passed by value, reference, or **`const`** reference, and whether the function should be a **`const`** function, **`static`** function, and/or **`friend`** function.
  2) **Stream insertion operator** (`"<<"`). A **`Matrix`** object can be written to **any** output stream using the stream insertion operator. Please refer to the **sample output files** to understand the required output format for a **`Matrix`** object. It is your responsibility to correctly determine the function return type, function argument(s), whether the argument(s) should be passed by value, reference, or **`const`** reference, and whether the function should be a **`const`** function, **`static`** function, and/or **`friend`** function.
- You **cannot** ~~add more data fields or functions~~ to the class. The above data fields and functions are sufficient to complete all the tasks in this assignment.

Other Development Notes

- You should use the **`'+'`** operator in the **`main()`** function to add two matrices together.
- Your program should smartly skip those empty lines in the input file.
- For all the functions, it is your responsibility to correctly determine the function return type, function argument(s), whether the argument(s) should be passed by value, reference, or **`const`** reference, and whether the function should be a **`const`** function, **`static`** function, and/or **`friend`** function.
- For all the functions, you **must** write **function docstrings** which include the description of function behavior, explanation of each function argument, and explanation of function return value. **Failing** to do this will result in **losing points**.

Sample Input and Output Files (Click to Download)

Sample Input File 1 (https://drive.google.com/uc?export=download&id=1d9JZPr3ac4KHJ9De0KOXyBj-acalODXb) **Sample Input File 2 (**
Sample Output File 1 (https://drive.google.com/uc?export=download&id=1yfMi719vyhLodcDuqwy80--e_fnFemFQ) **Sample Output File 2**

Assignment Submission and Grading (Please Read)

- Please upload all your **.h** (if any) and **.cpp** files (**not** the ~~entire Microsoft Visual Studio project folder~~) on Canvas.
- Before the assignment deadline, you can submit your work unlimited times. However, only your latest submission will be graded.
- At least **20**% of your code should be **comments**. All variable, function (if any), and class (if any) names should "make good sense". You should let the grader put **least effort** to understand your code. Grader will **take off points**, even if your program passes all test cases, if he/she has to put extra **unnecessary** effort to understand your code.
- Please **save a backup copy** of all your work in your computer hard drive.
- Your program will be graded (tested) using another valid input file (still named **matrices.txt**) to check whether it can generate the expected (correct) output file (with correct format and correct output values in it). As long as the input file is valid, your program should generate a correct output file. In other words, your program should work for **any** valid input file, **not** just the sample input files provided in the assignment instructions.
- In this class, you can assume that the input file (input data) is always **valid** and **has correct format**. You do **not** need to deal with ~~invalid input~~ or ~~error handling~~.
- Your work will be graded after the assignment deadline. All students will receive their assignment grades at (almost) the same time.