Sabanci University

Faculty of Engineering and Natural Sciences
CS204 Advanced Programming
Summer 2021-2022

THE 4 – The Narrow Way
Due: 2 September 2022 11.55pm (Sharp Deadline)

DISCLAIMER:

Only checking the sample run cases might not be sufficient as your solution will be checked against a variety of samples different from the provided samples; however checking these cases is highly encouraged and recommended.

You can NOT collaborate with your friends and discuss your solutions with each other. You have to write down the code on your own. Plagiarism will not be tolerated AND cooperation is not an excuse!

Introduction

The aim of this assignment is to practice on operator overloading. In this assignment, you will implement a well-structured *dynamic 2D matrix class* with several operators overloaded.

Main is Given (yeey!)

We have given you the *main.cpp* file that you will use in your project. You **are NOT allowed to change** this file (we will replace it while grading anyways :)). What you rather need to do is to implement your class in a way that it will work harmoniously with the given main function. Also we may test your submissions with a different main.cpp file, so you should implement the methods and operators in the way they should be, rather than finding workarounds. Please refer to *What and where to submit…* section for important detail.

What does Main do?

The main function reads two different file names from the console and opens them. Later, by using the extraction operator (>>) that you will overload, it will read the contents of each file into a dynamic 2D matrix object of type integer. Later, it does several operations with this constructed 2D matrix and displays them on the console using the insertion operator (<<) that

you will overload. You can inspect the main file for details, as it is quite short.

Class to Implement

We have given you an implementation of the Matrix2D class as a basis. This basis class:

1) Does not implement ***default*** constructor,
2) Does not implement ***deep copy*** constructor,
3) Does not implement ***destructor***, and
4) Lacks several other operators that the main function uses.

The ***definitive tasks*** of the assignment can be ordered like this:

- Implement isEmpty member function,
- Implement the default constructor (it may set everything to minimum default),
- Implement the destructor (deallocate the 2D dynamic array held within the object),
- Implement >> operator (we recommend you to implement this function at this stage, as it helps debugging other functions. You may assume that this function will only be used with ifstream objects, but not with cin or istringstream),
- Implement << operator,
- Implement += and = operators for the class,
- Implement the copy constructor (you will need this for + operator. Do not make shallow copy, it will change outputs in the main and make your program fail test cases), and
- Finally, implement + operator which will return a new Matrix2D object.

Format of the Input Files

There are very good assumptions that you can make on the format of the input files. These input files are the ones from which you will read the elements of the matrices. First of all, each row of the file represents a row of the matrix. The elements in a line can be separated by spaces, tabs or a combination of them, but this does not matter as you are already familiar with string streams.

The other assumption that you can make is that there will be an equal number of elements in each line of the file. Therefore, you do not need to check against this. Also, the number of lines in the file or the number of elements in a line will not ever be zero!

Lastly, the elements in the file will be whole numbers so that they can be parsed as integers.
In order to have pretty and neat outputs like we have in the sample runs, you will use setw(5) from the iomanip library.

Rules

In this assignment, the existence of *main.cpp* already narrows down the kind of implementation

that you can follow. On top of this, there are other limitations as well.

First of all, you are **not** allowed to use vectors or other data structures inside your class to hold the data. It **must** be a 2D dynamic array as it is in the basis class. These restrictions follow the fact that implementing the assignment operator, destructor and copy constructor would be meaningless with a vector field in the class (as these would be done automatically). We want you to get experience on implementing these methods, hence we restrict such usages.

Second of all, the definitive tasks given in the previous section **must** be all there. Although one or two of them may seem irrelevant to you, **you must** implement all of them or you will risk your partial credits from the assignment. We always inspect your codes and see what you lack or do not lack in your implementation.

Sample Runs

Below, we provide some sample runs of the program that you will develop. The *italic* and **bold** phrases are the standard inputs (cin) taken from the user (i.e., like ***this***). You have to display the required information in the same order and with the same words as here. Please refer to the submission page on SUCourse for additional sample runs.

Sample Run 1

```
file1.txt file2.txt
     2     4     6     3    10
     4     6     8     7    12
     6     8    -4     9    14
     8    10    20    14    16
    10    12    12    10    18


     2     4     6     3    10
     4     6     8     7    12
     6     8    -4     9    14
     8    10    20    14    16
    10    12    12    10    18

     3     6     9     2    15
     6     9    12     9    18
     9    12   -13    12    21
    12    15    34    21    24
    15    18    17    12    27

 10000     6     9     2    15
     6     9    12     9    18
     9    12   -13    12    21
```

```
    12    15    34    21    24
    15    18    17    12    27

3
```

Sample Run 2

file3.txt file4.txt

```
     3    23    18    -3     3    14     1
     8     7     6     5     4     3     2
    10    51     8    17   -48     5    64
     6    16     6    12     4     8     0
     5    27    30    -5     4    17     9
     5     4     3     1    -1    -3    -5
     6    44    11    23   -41     4    56
    -2     9     4    10    10     9     8


     3    23    18    -3     3    14     1
     8     7     6     5     4     3     2
    10    51     8    17   -48     5    64
     6    16     6    12     4     8     0
     5    27    30    -5     4    17     9
     5     4     3     1    -1    -3    -5
     6    44    11    23   -41     4    56
    -2     9     4    10    10     9     8

     3    44    41   -13     1    24     2
    15    13    11     9     7     5     3
    15    94    18    39   -90     8   119
    10    25     8    16     9    14     0
     5    48    53   -15     2    27    10
    12    10     8     5     2    -1    -4
    11    87    21    45   -83     7   111
     2    18     6    14    15    15     8

 10000    44    41   -13     1    24     2
    15    13    11     9     7     5     3
    15    94    18    39   -90     8   119
    10    25     8    16     9    14     0
     5    48    53   -15     2    27    10
    12    10     8     5     2    -1    -4
    11    87    21    45   -83     7   111
     2    18     6    14    15    15     8
```

Some Important Rules

In order to get full credit, your program must be efficient, modular (with the use of functions), well commented and properly indented. Besides, you also have to use understandable identifier names. Presence of any redundant computation, bad indentation, meaningless identifiers or missing/irrelevant comments may decrease your grade in case that we detect them. When we grade your THEs, we pay attention to these issues. Moreover, **we may test your programs with very large test cases**. Hence, take into consideration the efficiency of your algorithms other than correctness.

Sample runs give a good estimate of how correct your implementation is, however, we will test your programs with different test cases and **your final grade may conflict with what you have seen on CodeRunner**. We will also **manually** check your code, indentations and so on, hence do not object to your grade based on the **CodeRunner** results, but rather, consider every detail on this documentation. **So please make sure that you have read this documentation carefully and covered all possible cases, even some other cases you may not have seen on CodeRunner or the sample runs**. The cases that you *do not need* to consider are also given throughout this documentation.

Submit via SUCourse ONLY! Paper, e-mail or any other methods are not acceptable.

The internal clock of SUCourse might be a couple of minutes skewed, so make sure you do not leave the submission to the last minute. In the case of failing to submit your THE on time:

"No successful submission on SUCourse on time = A grade of zero (0) directly."

What and where to submit (PLEASE READ, IMPORTANT)

It'd be a good idea to write your name and last name in the program (as a comment line of course). Do not use any Turkish characters anywhere in your code (not even in comment parts). If your full name is "Duygu Karaoğlan Altop", and if you want to write it as comment; then you must type it as follows:

```
// Duygu Karaoglan Altop
```

You should update the attached header file and upload it into the attachment section under the "Answer" area in the relevant assignment submission page on SUCourse. **The name of the header file must be ("`"myUpdatedMatrixClass.h"`)**.

Please note that the warnings are also considered as errors on CodeRunner, which means that you should have a compiling and warning-free program.

Since the grading process will be automatic, you are expected to strictly follow these guidelines. If you do not follow these guidelines, your grade will be zero (0). Any tiny change in the output format will result in your grade being zero (0), so please test your programs yourself, and against the sample runs that are available at the relevant assignment submission page on SUCourse.

In the CodeRunner, there are some visible and invisible (hidden) test cases. You will see your final grade (including hidden test cases) before submitting your code. There is no re-submission. You don't have to complete your task in one time, you can continue from where you left last time but you should not press submit before finalizing it. Therefore, you should make sure that it's your final solution version before you submit it. Also, we still do not suggest that you develop your solution on CodeRunner but rather on your IDE on your computer.

You may visit the office hours if you have any questions regarding submissions.

How to get help?

You may ask your questions to TAs or to the instructor. Information regarding the office hours of the TAs and the instructor are available at SUCourse.

Plagiarism

Plagiarism is checked by automated tools, and we are very capable of detecting such cases. Be careful with that. Exchange of abstract ideas are totally okay but once you start sharing the code with each other, it is very probable to get caught by plagiarism. So, do NOT send any part of your code to your friends by any means or you might be charged as well, although you have done your THE by yourself. THEs are to be done personally and you have to submit your own work. **Cooperation will NOT be counted as an excuse.**

In case of plagiarism, the rules on the Syllabus apply.

Good Luck!
Ahmed Salem, Duygu Karaoğlan Altop