Sabanci University

Faculty of Engineering and Natural Sciences
CS204 Advanced Programming
Summer 2017-2018

Homework 4 – The Narrow Way
Due: 3 August 2018 11.55pm (Sharp Deadline)

DISCLAIMER:

Your program should be a robust one such that you have to consider all relevant user mistakes and extreme cases; you are expected to take actions accordingly!

Only checking the sample run cases might not be sufficient as your solution will be checked against a variety of samples different than the provided samples; however checking these cases are 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 homework is to practice on class templates and operator overloading. In this homework, you will implement a well-structured templated *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 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 homeworks 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 do not forget to change the name of *main.cpp* file to ***"SUCourseUserName_hw4.cpp"***. **This is very important; otherwise we cannot grade your homework**.

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 reads the contents of each file into two different dynamic 2D matrix objects of types integer and string. Later, it does several operations with these constructed 2D matrices 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) Is not templated,
2) Works for only integers,
3) Does not implement default constructor,
4) Does not implement deep copy constructor,
5) Does not implement destructor, and
6) Lacks several other operators that the main function uses.

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

- Convert the basis class into templated version,
- 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 main and make your program fail test cases.), and
- Finally, implement + operator which will return a new Matrix2D object.

As you all know, templated class function definitions should be in the same translation unit with the source files that they are used within. In order for you not to have any problems with this, and also for us to construct the projects for grading easier, please do all the implementation in the **header file** as given and do **not** create a .cpp file for the Matrix2D class.

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 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 some whole numbers so that they can be parsed as strings and integers at the same time.

In order to have pretty and neat outputs like we have in the sample runs, you will use setw(5) from the iomanip library. This way, it is better for our eyes while grading :)

Rules

In this homework, 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 homework. 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.

Sample Run 1

```
Please enter a file name: file1.txt
Please enter another file name: 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

   11   22   33  4-1   55
```

```
    22    33    44    52    66
    33    44   5-9    63    77
    44    55   614    77    88
    55    66    75    82    99

     2     4     6     8    10
     4     6     8    10    12
     6     8    10    12    14
     8    10    12    14    16
    10    12    14    16    18

    11    22    33   4-1    55
    22    33    44    52    66
    33    44   5-9    63    77
    44    55   614    77    88
    55    66    75    82    99

     3     6     9     7    15
     6     9    12    12    18
     9    12     1    15    21
    12    15    26    21    24
    15    18    19    18    27

 10000     6     9     7    15
     6     9    12    12    18
     9    12     1    15    21
    12    15    26    21    24
    15    18    19    18    27

3
Press any key to continue . . .
```

Sample Run 2

```
Please enter a file name: file3.txt
Please enter another file name: 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

 30   221  -523  7-10   5-2   410   001
 17    16    15    14    13    12    11
 55   843  -210  -522 -6-42    23   955
 24    79    42    84   -15    26    00
 50   621   723  5-10   6-2   710    81
-27   -26   -25   -34   -43   -52   -61
 15   143   110   122  1-42    13   155
-64    09    22    64    55    36    80

  6     4   -10    14    10     8     0
  2     2     2     2     2     2     2
 10    16    -4   -10   -12     4    18
  4    14     8    16    -2     4     0
 10    12    14    10    12    14    16
 -4    -4    -4    -6    -8   -10   -12
  2     2     2     2     2     2     2
-12     0     4    12    10     6    16

 30   221  -523  7-10   5-2   410   001
 17    16    15    14    13    12    11
 55   843  -210  -522 -6-42    23   955
 24    79    42    84   -15    26    00
 50   621   723  5-10   6-2   710    81
-27   -26   -25   -34   -43   -52   -61
 15   143   110   122  1-42    13   155
-64    09    22    64    55    36    80

  6    25    13     4     8    18     1
  9     8     7     6     5     4     3
 15    59     6    12   -54     7    73
  8    23    10    20     3    10     0
 10    33    37     0    10    24    17
  3     2     1    -2    -5    -8   -11
  7    45    12    24   -40     5    57
 -8     9     6    16    15    12    16

10000    25    13     4     8    18     1
```

```
 9     8     7     6     5     4     3
15    59     6    12   -54     7    73
 8    23    10    20     3    10     0
10    33    37     0    10    24    17
 3     2     1    -2    -5    -8   -11
 7    45    12    24   -40     5    57
-8     9     6    16    15    12    16

6
Press any key to continue . . .
```

Some Important Rules

Although some of the information is given below, please also read the homework submission and grading policies from the lecture notes of the first week. In order to get a full credit, your program must be efficient, modular (with the use of functions), well commented and 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 homeworks, we pay attention to these issues. Moreover, in order to observe the real performance of your codes, we are going to run your programs in Release mode and **we may test your programs with very large test cases**. Hence, take into consideration the efficiency of your algorithms other than correctness.

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.

WE WILL NOT USE GRADE CHECKER FOR THIS HOMEWORK!

As templated definitions are quite inconsistent across compilers, we will not use grade checker for this homework. Two text files are given for you to test your implementation and you can create more files yourselves for testing, as well. We will **not** use those input files during grading. Also, we will also use a different main file content during grading. But still, you **must** submit *main.cpp* by changing its name as instructed above.

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 homework on time:

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

What and where to submit (PLEASE READ, IMPORTANT)

You should prepare your program using MS Visual Studio 2012, as we will use it while testing your homework this time.

It'd be a good idea to write your name and lastname 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

Submission guidelines are below. Since the grading process will be semi-automatic, you are expected to strictly follow these guidelines. If you do not follow these guidelines, your grade will be zero. The lack of even one space character in the output will result in your grade being zero, so please test your programs yourself.

- Name the main.cpp into:

  "SUCourseUserName_hw4.cpp"

  Your SUCourse user name is actually your SUNet username which is used for checking sabanciuniv e-mails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name. For example, if your SU e-mail address is **atam@sabanciuniv.edu**, then the file name must be: **"atam_hw4.cpp"**.

- Please make sure that the files are the latest versions of your homework program.

- You should upload the sample txt files, the header you implemented and the main cpp file with its new name.

- Do not zip any of the documents but upload them as separate files only.

- Submit your work **through SUCourse only**!

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

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 homework by yourself. Homeworks 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!
Tolga Atam, Duygu K. Altop