**Sabanci University**

Faculty of Engineering and Natural Sciences

CS204 Advanced Programming

Summer 2018-2019

Homework 1 – Welcome to the Machine

Due: 9 July 2019 11.55pm (SHARP)

|  |
| --- |
| **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 must** 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 recall CS201 material and practice on matrices (i.e., two dimensional vectors), file streams and string operations. In this homework, you are going to implement an interpreter (something similar to a compiler), which parses the commands written in a custom command language (kind of a programming language) that is developed by us, and executes the matrix operations mandated by these commands.

**Inputs to Your Program**

The first input to your program is the name of a file, in which there will be a matrix, structured as described below. Therefore, your program should first prompt for a filename and read it from the standard input (cin). As mentioned, this file will have the elements of a matrix. If the input file cannot be opened successfully, then your program should ask for another filename, and try opening it repeatedly until the file is successfully opened.

While reading the file, **you must store the matrix in a vector of integer vectors**. Reading the file more than once would clearly be a bad solution and it will cause a grade deduction.

The second input to your program is the name of another file, in which there will be the commands for the matrix operations. Thus, your program should again prompt for a filename and read it from the standard input. As in the case with the first input, in case of a failure in opening the file, the filename should be asked repeatedly, until success.

**Format of the Inputs**

Both of the files start with a line indicating the number of rows and columns. These two pieces of information may be separated by space(s), tab(s), or a mixture of both. For the matrix file, these two numbers exactly tell how many rows and columns the matrix has. You can trust that the content of the file will obey this rule all the time. On the other hand, for the command file, these two numbers tell for which dimensionality of a matrix this file is written. Here, you can make sure that there will not be any commands working outside of these boundaries, in the remaining part of this file.

After the initial line, in the matrix file, there will be as many more lines as dictated by the row count, and each of these lines will have as many elements as dictated by the column count. Thus, each line represents a row of the matrix, and the elements on the lines make up the columns. Here, the elements on a line may be separated by space(s), tab(s), or a mixture of both. You may assume that the matrix file will consist only of digits and spaces/tabs/newlines/carriage-returns (“\r”). Also, the integers will not be negative.

On the other hand, the command file does not have a regular line structure after the initial line. Here are the rules: (i) each command in the command file ends with a semicolon (;), and (ii) in a command, there can be spaces, tabs, carriage-returns or even newlines (which means a command may cover multiple lines).

For both of the files, you may assume that no format issue will be present. Hence, you do not need to check the content or format of the file before parsing.

For any further details, you can see the sample runs and sample files.

**Program Flow and Commands**

Your program should start by asking the filenames for the matrix and commands files. While doing so, your program should make sure that the files with the given names actually exist. Later, your program should check if the two files are compatible with each other, meaning that your program should make sure that the command file is not written for a matrix with larger dimensions. If the files are compatible, then your program should carry on and process the commands. Otherwise, it should display an error message and stop its execution.

You may find the command language related information in the table given below.

|  |  |  |
| --- | --- | --- |
| **OPERATION** | **OPERATOR** | **DESCRIPTION** |
| *swap* | <> | swaps two elements, two rows or two columns of the matrix |
| *copy* | > | copies an element, row or column to another |
| *output* | o / O | displays an element, row, column or all of the matrix |
| *input* | i / I | takes a nonnegatıve integer as input and replaces an element, row, column or all of the matrix with this value |

Details of the above-given commands are as listed below:

1. *Swap* and *copy* operations take two operands: a left operand and a right operand.   
     
   Such commands will be in the form of "*a.b*", where *a* denotes the index of the row and *b* denotes the index of the column, separated by a single dot. Whenever the operation is to be carried out among the rows or columns, *x or X (case insensitive)* will be used to indicate this generality, as exemplified below. Please note that there is no option of having a command as *x.x*, which means that at least one of the operands has to be a numeric value.
   1. Operands may both represent the elements of the matrix:  
        
      4.2<>5.0 → swap the element at 4th row and 2nd column with the   
       element at 5th row and 0th column
   2. Operands may both represent the rows of the matrix:  
        
      2.x>5.x → copy the 2nd row to the 5th row
   3. Operands may both represent the columns of the matrix:

x.3<>x.0 → swap the 3rd column with the 0th column

1. *Input* and *output* operations take either one or zero operands. If there is an operand, then it represents an element, a row or a column of the matrix, and the syntax of this representation is the same as described above. On the other hand, if they do not take any operands, then the input/output operation deals with the whole matrix.
   1. i10.2 → input an integer and write it to the element at 10th row   
       and 2nd column
   2. o2.X → display the 2nd row entirely
   3. o → display the whole matrix
   4. i → input an integer and fill in the whole matrix with it

Please pay attention that, the parts of a command never needs any spacing to comply with the syntax. However, there still might be spaces, tabs and/or newlines between the operands and the operator; **anywhere within a command except for a number** (i.e. 1 0.2 is not a valid syntax as the number 10 is interleaved).

Another important point is that the characters "i", "o", and "x" that are used in the commands are case-insensitive. Your program should work correctly for both cases of these characters.

The command file that your program uses will never have syntax errors. While processing the command files, you do not need to check for syntactic errors. For example, "*1.x <> x.3*" is not possible in a command file as it mixes the row and column declarations. Also, there will not be any illegal characters in the command files (spaces, tabs and newlines are legal, and they do not hold any meaning).

While dealing with commands working on a row, column or the whole matrix, please be assured that the parts of the matrix that fall outside of the command file’s scope are not affected nor used. For example, given a 9x9 matrix and command file working with 6x6, the operation "o" should be printing the 6x6 submatrix, not the whole 9x9 matrix. Similarly, *swap*, *copy* and *input* operations should **never** edit the parts other than this submatrix that the commands file declares.

Your program should process these commands one by one and display messages about the commands processed. Once the command file is exhausted, your program should display that the command file is finished and it should print the whole original matrix. Please be reminded that this matrix may be different than what an "o" command from the command file would display, because the command file may have considered only a part of the main matrix if its declared dimension is lower than this main matrix.

**Sample Runs**

Below, we provide some sample runs of the program that you will develop. The *italic* and **bold** phrases are the standard input (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/spaces as here; in other words, there must be an exact match!

We will be automatically grading your homework using GradeChecker, so it is very important to satisfy the exact same output given in the sample runs. You can utilize GradeChecker [(http://sky.sabanciuniv.edu:8080/GradeChecker/)](http://sky.sabanciuniv.edu:8080/GradeChecker/) to check whether your code is working in the expected way. To be able to use GradeChecker, you should upload all of your files used in the homework **without zipping them**. Just a reminder, you will see a character ¶ which refers to a newline in your expected output.

**Sample Run 1**

Welcome to the first boring CS204 Homework.

This program does matrix operations as instructed by a command file.

In order to start, please enter a matrix file name: ***matrix1.txt***

Success!

Please enter a command file name: ***commands1.txt***

Success!

The matrix is being printed:

0 9 2 5 5 2 1 3 1

1 7 2 8 4 3 9 8 1

8 2 9 4 5 7 8 9 6

2 3 2 1 4 5 6 5 4

7 3 4 9 8 1 2 7 8

2 5 9 6 6 5 4 5 6

0 8 0 8 9 9 6 3 2

The element at Row 1, Column 1 is being swapped with the element at Row 3, Column 3.

Row 1 is being printed:

1 1 2 8 4 3 9 8 1

Row 0 is being swapped with Row 6.

Row 4 is being copied to Row 6.

Row 0 is being printed:

0 8 0 8 9 9 6 3 2

Row 4 is being printed:

7 3 4 9 8 1 2 7 8

Row 6 is being printed:

7 3 4 9 8 1 2 7 8

The matrix is being printed:

0 8 0 8 9 9 6 3 2

1 1 2 8 4 3 9 8 1

8 2 9 4 5 7 8 9 6

2 3 2 7 4 5 6 5 4

7 3 4 9 8 1 2 7 8

2 5 9 6 6 5 4 5 6

7 3 4 9 8 1 2 7 8

Column 5 is being swapped with Column 8.

Column 8 is being copied to Column 0.

Column 0 is being printed:

9 3 7 5 1 5 1

The matrix is being printed:

9 8 0 8 9 2 6 3 9

3 1 2 8 4 1 9 8 3

7 2 9 4 5 6 8 9 7

5 3 2 7 4 4 6 5 5

1 3 4 9 8 8 2 7 1

5 5 9 6 6 6 4 5 5

1 3 4 9 8 8 2 7 1

Commands are completed.

Printing the main matrix:

9 8 0 8 9 2 6 3 9

3 1 2 8 4 1 9 8 3

7 2 9 4 5 6 8 9 7

5 3 2 7 4 4 6 5 5

1 3 4 9 8 8 2 7 1

5 5 9 6 6 6 4 5 5

1 3 4 9 8 8 2 7 1

**Sample Run 2**

Welcome to the first boring CS204 Homework.

This program does matrix operations as instructed by a command file.

In order to start, please enter a matrix file name: ***mattrixx.txt***

Wrong file name!

Please enter another matrix file name: ***matrix.txt***

Wrong file name!

Please enter another matrix file name: ***matrix2.txt***

Success!

Please enter a command file name: ***commmmmands.txt***

Wrong file name!

Please enter another command file name: ***commands***

Wrong file name!

Please enter another command file name: ***commands1.txt***

Success!

Command and matrix dimensions do not work together. Program exiting...

**Sample Run 3**

Welcome to the first boring CS204 Homework.

This program does matrix operations as instructed by a command file.

In order to start, please enter a matrix file name: ***matrix2.txt***

Success!

Please enter a command file name: ***commands2.txt***

Success!

The matrix is being printed:

1 6 8 4 5

5 5 6 8 4

0 6 0 9 12

123 8 9 1 5

0 0 0 0 0

Please enter an input for the element at Row 0, Column 0: ***200***

The element at Row 1, Column 1 is being printed:

5

Row 0 is being copied to Row 1.

The matrix is being printed:

200 6 8 4 5

200 6 8 4 5

0 6 0 9 12

123 8 9 1 5

0 0 0 0 0

Please enter an input for Row 4: ***3***

The matrix is being printed:

200 6 8 4 5

200 6 8 4 5

0 6 0 9 12

123 8 9 1 5

3 3 3 3 3

Please enter an input for the matrix: ***4***

The matrix is being printed:

4 4 4 4 4

4 4 4 4 4

4 4 4 4 4

4 4 4 4 4

4 4 4 4 4

Commands are completed.

Printing the main matrix:

4 4 4 4 4 6

4 4 4 4 4 1

4 4 4 4 4 3

4 4 4 4 4 4

4 4 4 4 4 0

**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 code, 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 [Course Google Drive Folder](https://drive.google.com/drive/folders/1EozjiM4Ogau08Zd2gemSbNjzQxBj-Rma?usp=sharing).

**YOU SHOULD USE GRADE CHECKER FOR THIS HOMEWORK!**

You should use Grade Checker (<http://sky.sabanciuniv.edu:8080/GradeChecker/>) to check your expected grade. Just a reminder, you will see a character ¶ which refers to a newline in your expected output.

Make sure you upload the .txt files, too.

Grade Checker and the automated grading system use a different compiler than MS Visual Studio does. Hence, you should check the "***Common Errors***" page to see some extra situations to consider while doing your homework. If you do not consider these situations, you may get a lower score (even zero) even your program works correctly with MS Visual Studio.

***Common Errors Page***: <http://sky.sabanciuniv.edu:8080/GradeChecker/commonerrors.jsp>

Grade Checker can be pretty busy and unresponsive during the last day of the submission. Due to this fact, leaving the homework for the last day generally is not a good idea. You may wait for hours to test your homework or make an untested submission, sorrily..

Grade Checker and Sample Runs together give a good estimate of how correct your implementation is, however we may test your programs with different test cases and **your final grade may conflict with what you have seen on Grade Checker.** We will also **manually** check your code (comments, indentations and so on), hence do not object to your grade based on the Grade Checker 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 Grade Checker or Sample Runs**. The cases that you *do not need* to consider are also given throughout this documentation.

Submit via SUCourse ONLY! **Grade Checker is not considered as a submission**. Paper, e-mail or any other methods are not acceptable, either.

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 test your program using Grade Checker. We will use the same UNIX based C++ compiler that Grade Checker uses for grading your homework.

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 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 and with the Grade Checker tool explained above.

* Name your cpp file that contains your program as follows:

***"SUCourseUserName\_hw1.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\_hw1.cpp"**

* Please make sure that this file is the latest version of your homework program.
* You should upload all the .txt filesto SUCourse as well.
* Do not zip any of the documents but upload them as separate files only.
* Submit your work **through SUCourse only**! You can use the Grade Checker only to see if your program can produce the correct outputs both in the correct order and in the correct format. It will not be considered as the official submission. You must submit your work to SUCourse.

*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 Karaoğlan Altop