| **Department of Computer and Software Engineering – ITU** |
| --- |
| **SE200T: Data Structures and Algorithms** |

| **Course Instructor: Nadir Abbas** | **Dated:** |
| --- | --- |
| **Teaching Assistant: Azeen Ahmed** | **Semester: Fall 2024** |
| **Session: 2024-2028** | **Batch: BSCE2024** |

# **Assignment 1. Array Class using Dynamic Memory Allocation**

| **Name** | **Roll number** | **Obtained Marks/35** |
| --- | --- | --- |
|  |  |  |

Checked on: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Submission:**

• Email instructor or TA if there are any questions. You cannot look at others’ solutions or use others’ solutions, however, you can discuss it with each other. Plagiarism will be dealt with according to the course policy.

• Submission after due time will not be accepted.

**In this assignment you have to do following tasks:**

**Task 1:** Ensure that you have installed all three softwares in your personal computer (Github, Cygwin & CLion). Now, accept the assignment posted in classroom (e.g Google, LMS etc) and after accepting, clone the repository to your computer. Make sure you have logged into the github app with your account.

**Task 2:** Open Cygwin app, Move to your code directory with following command “cd <path\_of\_folder>”

<path\_of\_folder> can be automatically populated by dragging the folder and dropping it to the cygwin window.

Run the code through Cygwin, use command “make run”, to get the output of the code

**Task 3:** Solve the given problems, write code using **CLion** or any other IDE.

**Task 4:** Keep your code in the respective git cloned folder.

**Task 5:** Commit and Push the changes through the Github App

**Task 5**: Write the code in separate files **(as instructed**). Ensure that file names are in lowercase (e,g **main.cpp**).

**Task 6:** Run ‘**make run**’ to run C++ code

**Task 7:** Run ‘**make test**’ to test the C++ code

**Problem Statement: Matrix Operations**

In this assignment, you will implement a Matrix class in C++ that supports basic matrix operations. This class will manage a dynamic matrix and provide functionalities such as setting elements, displaying the matrix, computing determinants, inverses, and transposes, and multiplying matrices. You will also create a menu-driven interface to interact with the class.

Class Attributes:

Private Attributes:

* double \*\*mat - A pointer to a dynamically allocated 2D array that represents the matrix.
* int n - The size of the matrix (number of rows/columns).

Public Methods:

* Constructor:
  + Matrix(int size) - Initializes a matrix of size size x size with all elements set to 0.
* Destructor:
  + ~Matrix() - Cleans up dynamically allocated memory.
* Copy Constructor:
  + Matrix(const Matrix& other) - Initializes a new matrix as a copy of an existing matrix/
* Methods:
  + void setElement(int row, int col, double value) - Sets the value of a specific matrix element. Displays an error message if indices are out of bounds.
  + double getElement(int row, int col) const - Retrieves the value of a specific matrix element. Displays an error message if indices are out of bounds.
  + void display() const - Displays the matrix in a formatted manner.
  + double getDeterminant() - Computes and returns the determinant of the matrix.
  + Matrix\* getInverse() - Computes and returns the inverse of the matrix if it is invertible. Returns nullptr and displays an error message if the matrix is not invertible.
  + Matrix\* multiply(const Matrix& other) const - Multiplies the matrix with another matrix and returns the result.
  + Matrix\* getTranspose() - Computes and returns the transpose of the matrix.

Round off function:-

You can use this function to round off your matrix upto 2 decimal digits places

void roundOff(int digits){

        for(int i = 0; i < n; ++i){

            for(int j = 0; j < n; ++j){

                mat[i][j] = round(mat[i][j]\*100)/100;

            }

        }

    }

### Problem Statement: Matrix Inversion and Trace Calculation

In this assignment, you will implement a Matrix class that supports matrix inversion (for square matrices) and trace calculation. Your class will include the following functionalities:

* + Matrix Inversion: Compute the inverse of a square matrix, if it exists.
  + Matrix Trace Calculation: Compute the trace of a square matrix (i.e., the sum of its diagonal elements).

### Class Attributes

#### **Private Attributes:**

* + std::vector<std::vector<double>> mat - A matrix represented using a 2D vector.
  + int rows - The number of rows in the matrix.
  + int cols - The number of columns in the matrix.

#### **Public Methods**

##### Constructor:

* + Matrix(int r, int c) - Initializes a matrix with r rows and c columns with all elements set to 0. The matrix should be square (i.e., r == c), otherwise, an error message is displayed.

##### Destructor:

* + ~Matrix() - Cleans up any resources used by the matrix.

##### Copy Constructor:

* + Matrix(const Matrix& other) - Initializes a new matrix as a copy of an existing matrix.

##### Methods:

* + void setElement(int row, int col, double value) - Sets the value of a specific matrix element. Displays an error message if indices are out of bounds.
  + double getElement(int row, int col) const - Retrieves the value of a specific matrix element. Displays an error message if indices are out of bounds.
  + void display() const - Displays the matrix in a formatted manner.
  + Matrix getInverse() const - Computes and returns the inverse of the matrix. If the matrix is not invertible (i.e., determinant is 0), an error message is displayed.
  + double getTrace() const - Computes and returns the trace of the matrix.

### Implementation Details

#### **Matrix Initialization and Display:**

* + Use std::vector<std::vector<double>> to represent the matrix.
  + Implement display() to print the matrix to the console.

#### **Matrix Inversion:**

* Implement getInverse() to return the inverse of the matrix. This method should handle cases where the matrix is not invertible by displaying an appropriate error message.

#### **Matrix Trace Calculation:**

* Implement getTrace() to calculate the trace of the matrix. The trace is the sum of the elements on the main diagonal.

#### **Error Handling:**

* Ensure that methods handling matrix operations handle cases where the matrix is not square or not invertible.

### Menu-Driven Interface

Create a menu-driven interface that allows the user to:

1. Set Matrix Element: Enter values to set specific elements in the matrix.
2. Display Matrix: Print the current state of the matrix.
3. Compute and Display Matrix Inverse: Calculate and display the inverse of the matrix.
4. Compute and Display Matrix Trace: Calculate and display the trace of the matrix.
5. Exit: Terminate the program.

**Assessment Rubric for Assignment**

| **Performance metric** | **CLO** | **Able to complete the task over 80% (4-5)** | **Able to complete the task 50-80% (2-3)** | **Able to complete the task below 50% (0-1)** | **Marks** |
| --- | --- | --- | --- | --- | --- |
| 1. Realization of experiment | 3 | Executes without errors excellent user prompts, good use of symbols, spacing in output. Through testing has been completed. | Executes without errors, user prompts are understandable, minimum use of symbols or spacing in output. Some testing has been completed. | Does not execute due to syntax errors, runtime errors, user prompts are misleading or non- existent. No testing has been completed. |  |
| 1. Conducting experiment | 2 | Able to make changes and answered all questions. | Partially able to make changes and few incorrect answers. | Unable to make changes and answer all questions. |  |
| 1. Computer use | 4 | Document submission timely. | Document submission late. | Document submission not done. |  |
| 1. Teamwork | 4 | Actively engages and cooperates with other group member(s) in effective manner. | Cooperates with other group member(s) in a reasonable manner but conduct can be improved. | Distracts or discourages other group members from conducting the experiment |  |
| 1. Laboratory safety and disciplinary rules | 2 | Code comments are added and does help the reader to understand the code. | Code comments are added and does not help the reader to understand the code. | Code comments are not added. |  |
| 1. Data collection | 2 | Excellent use of white space, creatively organized work, excellent use of variables and constants, correct identifiers for constants, No line-wrap. | Includes name, and assignment, white space makes the program fairly easy to read. Title, organized work, good use of variables. | Poor use of white space (indentation, blank lines) making code hard to read, disorganized and messy. |  |
| 1. Data analysis | 3 | Solution is efficient, easy to understand, and maintain. | A logical solution that is easy to follow but it is not the most efficient. | A difficult and inefficient solution. |  |
| **Total (out of 35):** | | | | |  |