Ghulam Ishaq Khan Institute (GIKI)

Assignment # 3

Subject: Introduction to Computing & AI	**Course Code:** CS-101-A-Fall-24
Class: BS AI – 1st, **Batch:** 34 - Fall - 24	**Submission Deadline:** 15/Dec/2024-Sun (11:59-PM)
Instructor: M. Qasim Riaz, **TA:** Miss Amna Arooj	**Total Marks:** 70 (Marks are divided question wise)

Note (Read notes & instructions first)

- First of all, read the instructions and statements of each exercise/question carefully then write the solution.
- ***Upload separate C++ Code File for all questions:***
  - Create different file for each question & assignment
  - Name of each file should contain your roll number, assignment number & question number in a specific format.
  - For Example, if your roll number is 2024532 you are doing 2nd assignment and question no 5 then file name of your C++ file should be written as ---> 2024532_2_5.cpp (Similarly, create for each question)
  - Now upload all of these files at Microsoft teams.

CHEATING/COPY CASE or LATE SUBMISSION even 1 minute late will be graded as STRAIGHT ZERO MARKS.

Question: 1 – COVID 19 CONTROLLER (20 - Marks, CLO 3, PLO 1)

Amna and Iffat are best friends. Amna lives in a joint family system, while Iffat lives in a nuclear family system. Iffat often expresses her love for the joint family system, but she acknowledges the challenges Amna faces, especially during the COVID-19 pandemic.

Amna's family comprises 10 members living in a small 2-room house. The family members include Amna, her parents, 1 younger sister, 2 elder brothers, her grandparents, her paternal uncle, and aunty. Among them, 4 have tested positive for COVID-19, making it urgent to separate the patients from non-patients.

Meanwhile, Iffat, living in a nuclear family, wants to assist by inviting healthy members to temporarily stay at her house for safety. She decides to make a roster of who should stay with her based on their health status.

Your Tasks:

1. Design a 2-D array to represent the **family details** with the following structure:
   - Row 1: Names of family members
   - Row 2: COVID-19 status (e.g., "P" for Positive, "N" for Negative) - *(Unsorted 2D array shown below)*

```
{ {"Amna", "Father", "Mother", "Sister", "Brother1", "Brother2", "Grandfather", "Grandmother", "Uncle", "Aunt"},
{"P", "N", "P", "N", "P", "P", "N", "N", "N", "N"} }
```

2. Sort the family details using **Selection Sort** based on the COVID-19 status, ensuring all "P" (Positive) members are grouped first, followed by "N" (Negative) members. *(Sorted arrays should be as follows)*

```
{ {"Amna", "Father", "Mother", "Sister", "Brother1", "Brother2", "Grandfather", "Grandmother", "Uncle", "Aunt"},
{"P", "N", "P", "N", "P", "P", "N", "N", "N", "N"} }
```

3. Add another 2-D array for **Iffat's house** to temporarily store the details (names and COVID-19 status) of non-COVID members from Amna's family. *(Family members without COVID stayed at Iffat's house)*

```
{ {"Father", "Sister", "Grandfather", "Grandmother", "Uncle", "Aunt"}, {"N", "N", N", "N", "N", "N"} }
```

4. Create and utilize the following functions, use of pointers is not mandatory:
   - **DISPLAY()**: To display the family details or Iffat's house roster.
   - **SWAP()**: To swap rows during sorting.
   - **SELECTION_SORT()**: To sort the family details based on the COVID-19 status.
   - **TRANSFER_NON_COVID():** To transfer Non-COVID patients to Iffat's house.

Question: 2 –Matrix Solver (50 - Marks, CLO 3, PLO 1)

The purpose of this assignment is to strengthen your understanding of **2-D arrays**, **function design**, and the **mathematical operations on matrices** in C++. By completing this assignment, you will learn how to implement matrix operations programmatically and apply core C++ concepts such as arrays, functions, and user input handling in arrays. ***Note:** In this assignment you will just work on 2x2 and 3x3 matrix)*

Instructions and Tasks

Step 1: User Interaction

1. Write a program that presents a **menu of options** to the user for matrix operations. The menu should include the following options:
    - Identify the type of matrix (e.g., square, scalar, diagonal, identity).
    - Identify the order of the matrix (e.g., 2x2, 3x3, etc.).
    - Find the transpose of a matrix.
    - Find the adjoint of a matrix.
    - Find the inverse of a matrix.
    - Multiply two matrices.
2. Implement a function to **display the menu** and prompt the user to select an operation (for example: Press 1 for finding transpose, press 2 for finding adjoint …. Similarly for all operations).
3. Validate the user's input and ensure only valid options can be selected.

Step 2: Matrix Creation

1. Write a function to **create a matrix or matrices** as required by the operation selected by the user.
    - For operations like transpose, inverse, adjoint, or type identification prompt the user to select the dimensions (2x2 or 3x3) of a single matrix.
    - For matrix multiplication, prompt the user to select the dimensions of both matrices and validate that the multiplication rule (columns of the first matrix must equal rows of the second matrix) is satisfied.
2. Populate the matrix (or matrices) by taking **element-wise input** from the user.

Step 3: Perform Matrix Operations

For each operation, implement a dedicated function to handle the calculations. Your program should execute the function corresponding to the user's choice.

(a) void checkMatrixType(); *//function to Identify the Type of Matrix – pass 2-D array as reference for matrix*

Write a function to determine the type of matrix and display it on screen. Use the following criteria:

- **Square Matrix**: The number of rows equals the number of columns.
- **Scalar Matrix**: A square matrix where all diagonal elements are the same, and all off-diagonal elements are zero.
- **Identity Matrix**: A square matrix where all diagonal elements are 1, and all off-diagonal elements are zero.
- **Diagonal Matrix**: A square matrix where all elements except the diagonal are zero.
- **Zero Matrix**: A matrix where all elements are 0.

(b) void checkMatrixOrder(); *//function to Find the Order of the Matrix (pass by reference)*

Write a function to return the order of the matrix in the format **rows x columns** and display it on screen.

(c) void calculateMatrixTranspose(); *//function to Find the Transpose of the Matrix (pass by reference)*

Write a function to compute the transpose of a matrix and display it on screen.

- The transpose is obtained by swapping rows and columns of the matrix.

(d) void calculateMatrixAdjoint(); *//function to Find the adjoint of the Matrix (pass by reference)*

Write a function to compute the adjoint of a matrix and display it on screen.

[illegible]

- The adjoint is the transpose of the cofactor matrix. Each cofactor is calculated by excluding the current row and column and finding the determinant of the resulting smaller matrix.

(e) void calculateMatrixInverse(); *//function to Find the Inverse of the Matrix (pass by reference)*

Write a function to calculate the inverse of a matrix and display it on screen.

- For a 2x2 matrix:
- For larger matrices, use the adjoint and determinant.
- Validate that the determinant is not 0 before calculating the inverse.

(f) void multiplyMatrices() *//function to multiply two Matrices (pass both by reference)*

Write a function to multiply two matrices and display it on screen.

- Ensure that the number of columns in the first matrix matches the number of rows in the second matrix.
- Perform the multiplication operation and return the resulting matrix.

Step 4: Repeat Operations

1. After performing an operation, ask the user if they want to:
    - Perform another operation on the same matrix/matrices if user wants.
    - Or create a new matrix/matrix for a different operation if the user wants.
2. You guys may Implement a do while – loop using switch statement inside of it to allow the program to handle multiple operations until the user decides to exit from program.

---

o --- | --- Good Luck --- | --- o

[illegible]