# Lab Manual 9:

**Name:** Romaisa Yaqoob

**ID:** 469297

**Class:** ME15 A

**Subject:** Fundamentals  of Programming Lab

**Date:** 12 Dec 2023

**Submitted to**: Sir Affan

Lab Tasks:

Task 1: Make 2D Array in C++ and print left diagonal and right diagonal sum of a 3x3 matrix.

Code:

```cpp
#include <iostream>
using namespace std;

int main() {
    const int size = 3;
    int matrix[size][size];

    // Input matrix elements from the user
    cout << "Enter the elements of the 3x3 matrix:" << endl;
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
            cout << "Element (" << i + 1 << ", " << j + 1 << "): ";
            cin >> matrix[i][j];
        }
    }

    // Display the entered matrix
    cout << "\nEntered Matrix:" << endl;
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
            cout << matrix[i][j] << " ";
        }
        cout << endl;
    }

    // Calculate and print the left diagonal sum
    int leftDiagonalSum = 0;
    for (int i = 0; i < size; ++i) {
        leftDiagonalSum += matrix[i][i];
    }
    cout << "\nLeft Diagonal Sum: " << leftDiagonalSum << endl;

    // Calculate and print the right diagonal sum
    int rightDiagonalSum = 0;
    for (int i = 0; i < size; ++i) {
        rightDiagonalSum += matrix[i][size - 1 - i];
    }
    cout << "Right Diagonal Sum: " << rightDiagonalSum << endl;

    return 0;
}
```

Output:

```
Enter the elements of the 3x3 matrix:
Element (1, 1): 8
Element (1, 2): -9
Element (1, 3): 2
Element (2, 1): 4
Element (2, 2): 5
Element (2, 3): 1
Element (3, 1): 0
Element (3, 2): -3
Element (3, 3): 6

Entered Matrix:
8 -9 2
4 5 1
0 -3 6

Left Diagonal Sum: 19
Right Diagonal Sum: 7
```

Task 2:Write a function to add two 2D arrays of size 3x3.
Code:

```cpp
#include <iostream>
using namespace std;

// Function to add two 3x3 matrices
void addMatrices(int matrix1[3][3], int matrix2[3][3], int result[3][3]) {
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            result[i][j] = matrix1[i][j] + matrix2[i][j];
        }
    }
}
// Function to display a 3x3 matrix
void displayMatrix(int matrix[3][3]) {
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << matrix[i][j] << " ";
        }
        cout << endl;
    }
}
int main() {
    int matrix1[3][3];
    int matrix2[3][3];
    int result[3][3];

    // Input elements for the first matrix
    cout << "Enter the elements of the first 3x3 matrix:" << endl;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << "Element (" << i + 1 << ", " << j + 1 << "): ";
            cin >> matrix1[i][j];
        }
    }
    // Input elements for the second matrix
    cout << "\nEnter the elements of the second 3x3 matrix:" << endl;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << "Element (" << i + 1 << ", " << j + 1 << "): ";
            cin >> matrix2[i][j];
        }
    }
    // Add matrices and display the result
    addMatrices(matrix1, matrix2, result);

    cout << "\nResultant Matrix (Sum of Matrices):" << endl;
    displayMatrix(result);
    return 0;
}
```

Output:

```
Enter the elements of the first 3x3 matrix:
Element (1, 1): 6
Element (1, 2): 3
Element (1, 3): 8
Element (2, 1): 3
Element (2, 2): 7
Element (2, 3): 5
Element (3, 1): 6
Element (3, 2): 0
Element (3, 3): 4

Enter the elements of the second 3x3 matrix:
Element (1, 1): -4
Element (1, 2): -2
Element (1, 3): 2
Element (2, 1): 5
Element (2, 2): 2
Element (2, 3): 0
Element (3, 1): 7
Element (3, 2): 6
Element (3, 3): 5

Resultant Matrix (Sum of Matrices):
2 1 10
8 9 5
13 6 9
```

Task 3: Using 2D arrays in C++, take transpose of a 3x3 matrix. Make a transpose function

Code:

```cpp
#include <iostream>
using namespace std;

// Function to calculate the transpose of a 3x3 matrix
void transposeMatrix(int matrix[3][3], int result[3][3]) {
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            result[j][i] = matrix[i][j];
        }
    }
}
// Function to display a 3x3 matrix
void displayMatrix(int matrix[3][3]) {
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << matrix[i][j] << " ";
        }
        cout << endl;
    }
}
int main() {
    int matrix[3][3];
    int transposeResult[3][3];

    // Input elements for the matrix
    cout << "Enter the elements of the 3x3 matrix:" << endl;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << "Element (" << i + 1 << ", " << j + 1 << "): ";
            cin >> matrix[i][j];
        }
    }
    // Calculate transpose and display the result
    transposeMatrix(matrix, transposeResult);

    cout << "\nOriginal Matrix:" << endl;
    displayMatrix(matrix);

    cout << "\nTransposed Matrix:" << endl;
    displayMatrix(transposeResult);

    return 0;
}
```

Output:

```
Enter the elements of the 3x3 matrix:
Element (1, 1): 6
Element (1, 2): 8
Element (1, 3): 3
Element (2, 1): 3
Element (2, 2): 9
Element (2, 3): 7
Element (3, 1): 2
Element (3, 2): 3
Element (3, 3): 4

Original Matrix:
6 8 3
3 9 7
2 3 4

Transposed Matrix:
6 3 2
8 9 3
3 7 4
```

Task 4: Using 2D arrays in C++, implement 3x3 matrix multiplication. Make a function.

Code:

```cpp
#include <iostream>
using namespace std;

// Function to perform matrix multiplication for 3x3 matrices
void multiplyMatrices(int matrix1[3][3], int matrix2[3][3], int result[3][3]) {
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            result[i][j] = 0;
            for (int k = 0; k < 3; ++k) {
                result[i][j] += matrix1[i][k] * matrix2[k][j];
            }
        }
    }
}
// Function to display a 3x3 matrix
void displayMatrix(int matrix[3][3]) {
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << matrix[i][j] << " ";
        }
        cout << endl;
    }
}
int main() {
    int matrix1[3][3];
    int matrix2[3][3];
    int productResult[3][3];

    // Input elements for the first matrix
    cout << "Enter the elements of the first 3x3 matrix:" << endl;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << "Element (" << i + 1 << ", " << j + 1 << "): ";
            cin >> matrix1[i][j];
        }
    }
    // Input elements for the second matrix
    cout << "\nEnter the elements of the second 3x3 matrix:" << endl;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << "Element (" << i + 1 << ", " << j + 1 << "): ";
            cin >> matrix2[i][j];
        }
    }
    // Perform matrix multiplication and display the result
    multiplyMatrices(matrix1, matrix2, productResult);

    cout << "\nResultant Matrix (Product of Matrices):" << endl;
    displayMatrix(productResult);
    return 0;
}
```

Output:

```
Enter the elements of the first 3x3 matrix:
Element (1, 1): 9
Element (1, 2): 8
Element (1, 3): 7
Element (2, 1): 6
Element (2, 2): 5
Element (2, 3): 4
Element (3, 1): 3
Element (3, 2): 2
Element (3, 3): 1

Enter the elements of the second 3x3 matrix:
Element (1, 1): -4
Element (1, 2): 3
Element (1, 3): -2
Element (2, 1): 8
Element (2, 2): -1
Element (2, 3): 3
Element (3, 1): 7
Element (3, 2): -3
Element (3, 3): 0

Resultant Matrix (Product of Matrices):
77 -2 6
44 1 3
11 4 0
```

q

Task 5: Print the multiplication table of 15 using recursion.

Code:

```cpp
#include <iostream>
using namespace std;

// Recursive function to print the multiplication table of 15
void printMultiplicationTable(int number, int multiplier, int limit) {
    if (multiplier > limit) {
        return;
    }

    cout << number << " * " << multiplier << " = " << (number * multiplier) << endl;

    // Recursive call with the next multiplier
    printMultiplicationTable(number, multiplier + 1, limit);
}

int main() {
    int number = 15;
    int limit;

    cout << "Enter the limit for the multiplication table: ";
    cin >> limit;

    cout << "Multiplication Table of " << number << " up to " << limit << ":" << endl;

    // Initial call to the recursive function
    printMultiplicationTable(number, 1, limit);

    return 0;
}
```

Output:

```
Enter the limit for the multiplication table: 10
Multiplication Table of 15 up to 10:
15 * 1 = 15
15 * 2 = 30
15 * 3 = 45
15 * 4 = 60
15 * 5 = 75
15 * 6 = 90
15 * 7 = 105
15 * 8 = 120
15 * 9 = 135
15 * 10 = 150

--------------------------------
Process exited after 2.663 seconds with return value 0
Press any key to continue . . .
```

## Home Task:

Task: Write a C++ program to take inverse of a 3x3 matrix using its determinant and adjoint

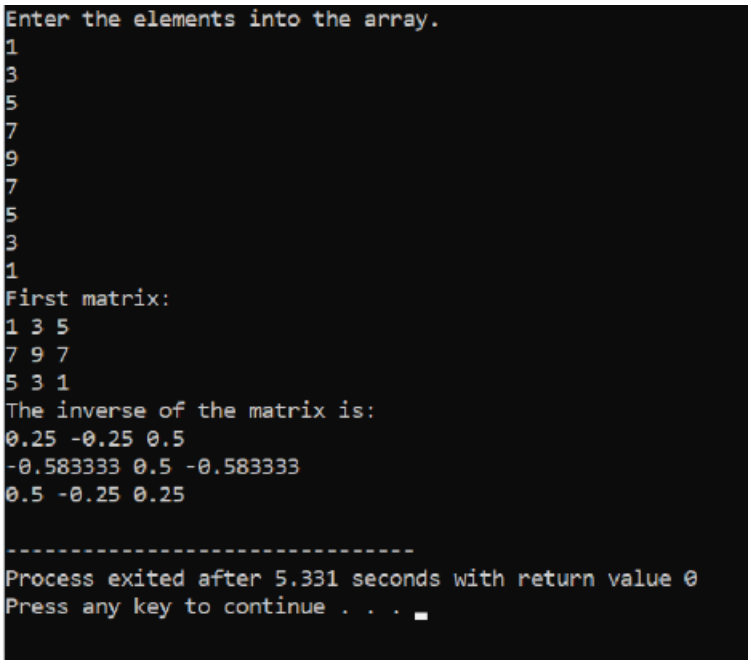
Code:

```
    float adjoint[3][3],inverse[3][3];
    array1(matrix1);
    cout<<"First matrix: \n";
    array1display(matrix1);
    float determinant = matrix1[0][0] * (matrix1[1][1] * matrix1[2][2] - matrix1[2][1] * matrix1[1][2]) -
                        matrix1[0][1] * (matrix1[1][0] * matrix1[2][2] - matrix1[2][0] * matrix1[1][2]) +
                        matrix1[0][2] * (matrix1[1][0] * matrix1[2][1] - matrix1[2][0] * matrix1[1][1]);
    if (determinant == 0) {
    cout << "The matrix is singular, it\'s inverse does not exist." << endl;
    }
    else{
        for(int i=0; i<3; i++){
            for(int j=0; j<3; j++){
                adjoint[i][j] = (matrix1[(j+1)%3][(i+1)%3] * matrix1[(j+2)%3][(i+2)%3] -
                matrix1[(j+1)%3][(i+2)%3] * matrix1[(j+2)%3][(i+1)%3]);
            }
        }


    for (int i = 0; i < 3; ++i){
        for (int j = 0; j < 3; ++j){
            inverse[i][j] = adjoint[i][j] / determinant;
        }
    }
    cout << "The inverse of the matrix is:" << endl;
    for (int i=0; i<3; i++) {
        for (int j=0; j<3; j++){
            cout << inverse[i][j] << " ";
        }
        cout << endl;
    }

}
```

Output 1:




Output 2: