# FUNDAMENTALS OF PROGRAMMING

## LAB MANUAL # 09

LAB INSTRUCTOR: MUHAMMAD AFFAN

STUDENT NAME: BUSHRA FAROOQ

CMS ID: 479973

DATE: 14/12/2023

# LAB TASK

## QUESTION 1

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

```
#include <iostream>

using namespace std;

int main() {

    const int size = 3;

    int matrix[size][size];



    cout << "Enter elements of the 3x3 matrix:\n";

    for (int i = 0; i < size; ++i) {

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

            cout << "Enter element at position (" << i + 1 << ", " << j + 1 << "): ";

            cin >> matrix[i][j];

        }

    }



    cout << "\nThe matrix is:\n";

    for (int i = 0; i < size; ++i) {

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

            cout << matrix[i][j] << " ";

        }

        cout << "\n";
```

```
    }


    int leftDiagonalSum = 0;
    for (int i = 0; i < size; ++i) {
        leftDiagonalSum += matrix[i][i];
    }
    cout << "\nLeft Diagonal Sum: " << leftDiagonalSum << "\n";


    int rightDiagonalSum = 0;
    for (int i = 0; i < size; ++i) {
        rightDiagonalSum += matrix[i][size - 1 - i];
    }
    cout << "Right Diagonal Sum: " << rightDiagonalSum << "\n";

    return 0;
}
```

```
#include <iostream>
using namespace std;
int main() {
    const int size = 3;
    int matrix[size][size];


    cout << "Enter elements of the 3x3 matrix:\n";
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
            cout << "Enter element at position (" << i + 1 << "
                , " << j + 1 << "): ";
            cin >> matrix[i][j];
        }
    }


    cout << "\nThe matrix is:\n";
    for (int i = 0; i < size; ++i) {
```

```
/tmp/FtelTWoteU.o
Enter elements of the 3x3 matrix:
Enter element at position (1, 1): 1
Enter element at position (1, 2): 2
Enter element at position (1, 3): 3
Enter element at position (2, 1): 4
Enter element at position (2, 2): 5
Enter element at position (2, 3): 6
Enter element at position (3, 1): 7
Enter element at position (3, 2): 8
Enter element at position (3, 3): 9
The matrix is:
1 2 3
4 5 6
7 8 9

Left Diagonal Sum: 15
Right Diagonal Sum: 15
```

## QUESTION 2

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

```
#include <iostream>

using namespace std;

void addMatrices(const int mat1[3][3], const int mat2[3][3], int result[3][3]) {
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            result[i][j] = mat1[i][j] + mat2[i][j];
        }
    }
}


int main() {
    const int size = 3;
    int matrix1[size][size] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    int matrix2[size][size] = {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}};
    int result[size][size];



    addMatrices(matrix1, matrix2, result);



    cout << "Matrix after addition:\n";
    for (int i = 0; i < size; ++i) {
```

```
        for (int j = 0; j < size; ++j) {
            cout << result[i][j] << " ";
        }
        cout << "\n";
    }

    return 0;
}
```

```
#include <iostream>
using namespace std;
void addMatrices(const int mat1[3][3], const int mat2[3][3],
    int result[3][3]) {
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            result[i][j] = mat1[i][j] + mat2[i][j];
        }
    }
}

int main() {
    const int size = 3;
    int matrix1[size][size] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}
    ;
    int matrix2[size][size] = {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}}
    ;
    int result[size][size];
```

```
/tmp/mn3hfPNk1a.o
Matrix after addition:
10 10 10
10 10 10
10 10 10
```

## QUESTION 3

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

#include <iostream>

using namespace std;

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];
        }
    }
}

int main() {
    int originalMatrix[3][3] = {{1, 4, 3}, {4, 9, 6}, {0, 8, 9}};
    int transposedMatrix[3][3];

    transposeMatrix(originalMatrix, transposedMatrix);

    cout << "Original Matrix:" << endl;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << originalMatrix[i][j] << " ";
        }
        cout << endl;
    }

    cout << "\nTransposed Matrix:" << endl;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << transposedMatrix[i][j] << " ";
        }
```

```
        cout << endl;
    }

    return 0;
}
```

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

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];
        }
    }
}

int main() {
    int originalMatrix[3][3] = {{1, 4, 3}, {4, 9, 6}, {0, 8, 9}};
    int transposedMatrix[3][3];

    transposeMatrix(originalMatrix, transposedMatrix);

    cout << "Original Matrix:" << endl;
```

```
/tmp/cxMup39GxS.o
Original Matrix:
1 4 3
4 9 6
0 8 9

Transposed Matrix:
1 4 0
4 9 8
3 6 9
```

## QUESTION 4

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

#include <iostream>

using namespace std;

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];
            }
        }
    }
}

int main() {
    int matrix_a[3][3] = {{1, 7, 3}, {4, 8, 6}, {7, 5, 9}};
    int matrix_b[3][3] = {{9, 3, 7}, {6, 4, 4}, {3, 9, 1}};
    int result_matrix[3][3];

    multiplyMatrices(matrix_a, matrix_b, result_matrix);

    cout << "Matrix A:" << endl;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << matrix_a[i][j] << " ";
        }
        cout << endl;
    }

    cout << "\nMatrix B:" << endl;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << matrix_b[i][j] << " ";
```

```
        }
        cout << endl;
    }

    cout << "\nResultant Matrix:" << endl;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << result_matrix[i][j] << " ";
        }
        cout << endl;
    }

    return 0;
}
```

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

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];
            }
        }
    }
}

int main() {
    int matrix_a[3][3] = {{1, 7, 3}, {4, 8, 6}, {7, 5, 9}};
    int matrix_b[3][3] = {{9, 3, 7}, {6, 4, 4}, {3, 9, 1}};
    int result_matrix[3][3];
```

```
/tmp/cxMup39GxS.o
Matrix A:
1 7 3
4 8 6
7 5 9

Matrix B:
9 3 7
6 4 4
3 9 1

Resultant Matrix:
60 58 38
102 98 66
120 122 78
```

## QUESTION 5

Print the multiplication table of 15 using recursion.

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

void printMultiplicationTable(int number, int multiplier) {
    if (multiplier <= 10) {
        cout << number << " x " << multiplier << " = " << number * multiplier << endl;
        printMultiplicationTable(number, multiplier + 1);
    }
}

int main() {
    int tableNumber = 15;

    cout << "Multiplication Table of " << tableNumber << ":" << endl;
    printMultiplicationTable(tableNumber, 1);

    return 0;
}
```

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

void printMultiplicationTable(int number, int multiplier) {
    if (multiplier <= 10) {
        cout << number << " x " << multiplier << " = " <<
            number * multiplier << endl;
        printMultiplicationTable(number, multiplier + 1);
    }
}

int main() {
    int tableNumber = 15;

    cout << "Multiplication Table of " << tableNumber << ":" <<
        endl;
    printMultiplicationTable(tableNumber, 1);
```

```
/tmp/cxMup39GxS.o
Multiplication Table of 15:
15 x 1 = 15
15 x 2 = 30
15 x 3 = 45
15 x 4 = 60
15 x 5 = 75
15 x 6 = 90
15 x 7 = 105
15 x 8 = 120
15 x 9 = 135
15 x 10 = 150
```

# HOME TASK

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

#include<iostream>

#include<cmath>

using namespace std;

float determinant2x2(float a, float b, float c, float d) {

    return a * d - b * c;

}

```
float determinant3x3(float matrix[3][3]) {
    return matrix[0][0] * determinant2x2(matrix[1][1], matrix[1][2], matrix[2][1], matrix[2][2])
-
           matrix[0][1] * determinant2x2(matrix[1][0], matrix[1][2], matrix[2][0], matrix[2][2]) +
           matrix[0][2] * determinant2x2(matrix[1][0], matrix[1][1], matrix[2][0], matrix[2][1]);
}

// Function to calculate the adjoint of a 3x3 matrix
void adjointMatrix(float matrix[3][3], float adjoint[3][3]) {
    adjoint[0][0] = determinant2x2(matrix[1][1], matrix[1][2], matrix[2][1], matrix[2][2]);
    adjoint[0][1] = -determinant2x2(matrix[1][0], matrix[1][2], matrix[2][0], matrix[2][2]);
    adjoint[0][2] = determinant2x2(matrix[1][0], matrix[1][1], matrix[2][0], matrix[2][1]);

    adjoint[1][0] = -determinant2x2(matrix[0][1], matrix[0][2], matrix[2][1], matrix[2][2]);
    adjoint[1][1] = determinant2x2(matrix[0][0], matrix[0][2], matrix[2][0], matrix[2][2]);
    adjoint[1][2] = -determinant2x2(matrix[0][0], matrix[0][1], matrix[2][0], matrix[2][1]);

    adjoint[2][0] = determinant2x2(matrix[0][1], matrix[0][2], matrix[1][1], matrix[1][2]);
    adjoint[2][1] = -determinant2x2(matrix[0][0], matrix[0][2], matrix[1][0], matrix[1][2]);
    adjoint[2][2] = determinant2x2(matrix[0][0], matrix[0][1], matrix[1][0], matrix[1][1]);
}

// Function to calculate the inverse of a 3x3 matrix
```

```
void inverseMatrix(float matrix[3][3], float inverse[3][3]) {
    float det = determinant3x3(matrix);

    if (det == 0) {
        cout << "Inverse does not exist as the determinant is zero." << endl;
        return;
    }

    float adjoint[3][3];
    adjointMatrix(matrix, adjoint);

    for (int i = 0; i < 3; ++i)
        for (int j = 0; j < 3; ++j)
            inverse[i][j] = adjoint[i][j] / det;
}

// Function to display a matrix
void displayMatrix(float 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() {
    float matrix[3][3];

    cout << "Enter the elements of the 3x3 matrix:" << endl;
    for (int i = 0; i < 3; ++i)
        for (int j = 0; j < 3; ++j)
            cin >> matrix[i][j];

    float inverse[3][3];
    inverseMatrix(matrix, inverse);

    cout << "Inverse matrix:" << endl;
    displayMatrix(inverse);

    return 0;
}
```

```cpp

#include<iostream>
#include<cmath>

using namespace std;


float determinant2x2(float a, float b, float c, float d) {
    return a * d - b * c;
}


float determinant3x3(float matrix[3][3]) {
    return matrix[0][0] * determinant2x2(matrix[1][1], matrix[1][2], matrix[2][1], matrix[2][2]) -
           matrix[0][1] * determinant2x2(matrix[1][0], matrix[1][2], matrix[2][0], matrix[2][2]) +
           matrix[0][2] * determinant2x2(matrix[1][0], matrix[1][1], matrix[2][0], matrix[2][1]);
```

```
/tmp/pIb8rYeEfs.o
Enter the elements of the 3x3 matrix:
1
0
5
2
1
6
3
4
0
Inverse matrix:
-24 18 5
20 -15 -4
-5 4 1
```