School Of Mechanical & Manufacturing Engineering, NUST

Department of Mechanical Engineering

CS-114 - Fundamentals of Programming

Lab Report # 09

Course Instructor: Dr Jawad Khan

Lab Instructor: Mr. Muhammad Affan, Mr. Saqib

Student Name: Ayesha Khan
CMS ID: 478212

DATE:
12-12-23

School Of Mechanical & Manufacturing Engineering, NUST

Department of Mechanical Engineering

Lab Report # 09
Functions & Multidimensional Arrays

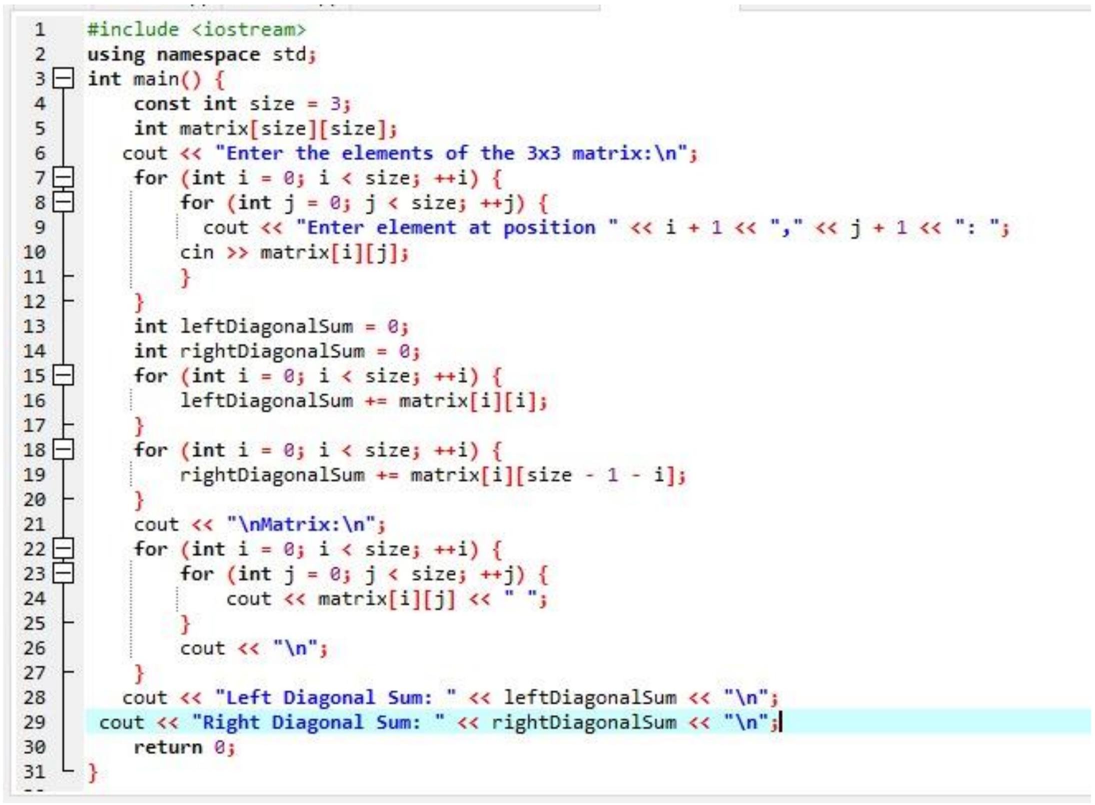
Objectives:
The objectives of this lab are:

➢to learn about functions, return values, and recursion

➢ to understand the use of 2D arrays.

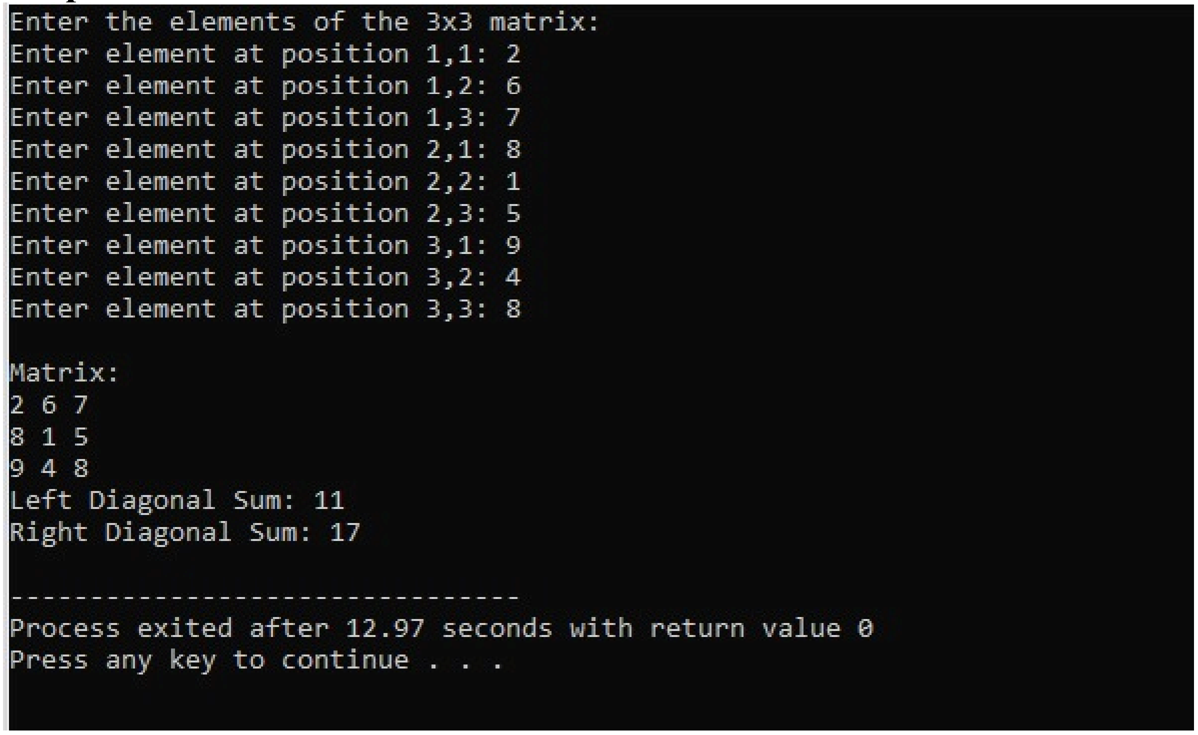
Lab Tasks:
Task 1:
Make 2D Array in C++ and print left diagonal and right diagonal sum of a 3x3 matrix.

Code:



```
#include <iostream>
using namespace std;
int main() {
    const int size = 3;
    int matrix[size][size];
   cout << "Enter the 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];
        }
    }
    int leftDiagonalSum = 0;
    int rightDiagonalSum = 0;
    for (int i = 0; i < size; ++i) {
        leftDiagonalSum += matrix[i][i];
    }
    for (int i = 0; i < size; ++i) {
        rightDiagonalSum += matrix[i][size - 1 - i];
    }
    cout << "\nMatrix:\n";
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
            cout << matrix[i][j] << " ";
        }
        cout << "\n";
    }
   cout << "Left Diagonal Sum: " << leftDiagonalSum << "\n";
 cout << "Right Diagonal Sum: " << rightDiagonalSum << "\n";
    return 0;
}
```

Output:

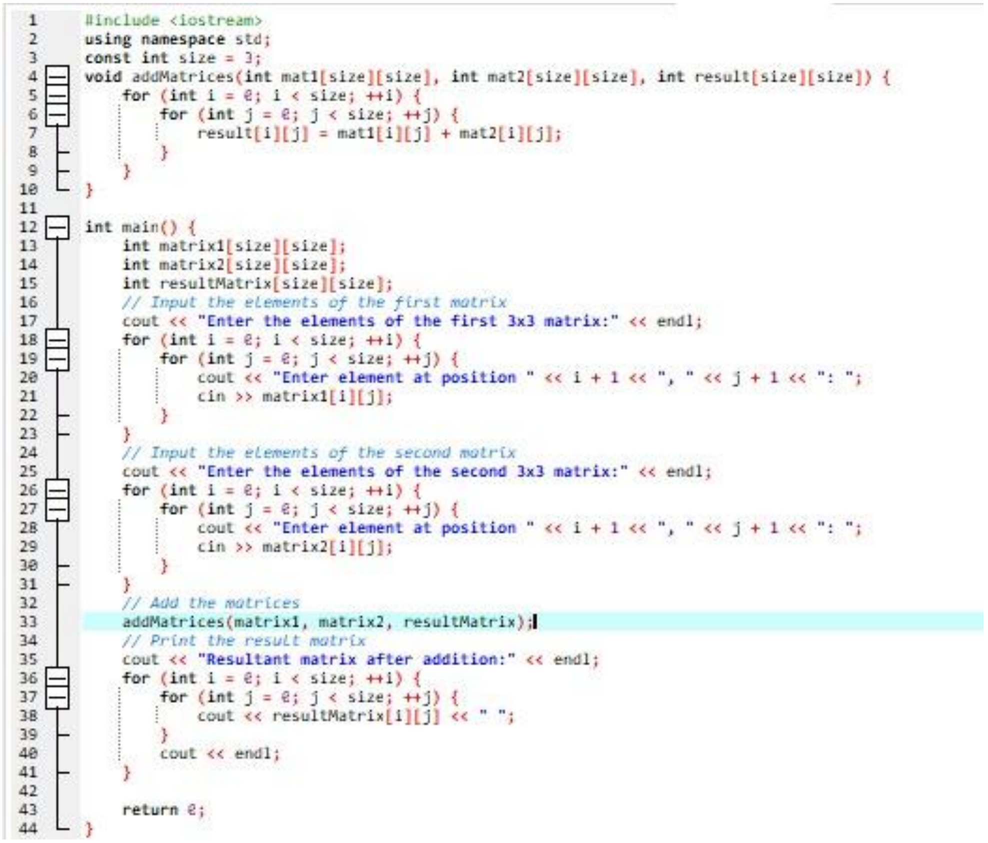


```
Enter the elements of the 3x3 matrix:
Enter element at position 1,1: 2
Enter element at position 1,2: 6
Enter element at position 1,3: 7
Enter element at position 2,1: 8
Enter element at position 2,2: 1
Enter element at position 2,3: 5
Enter element at position 3,1: 9
Enter element at position 3,2: 4
Enter element at position 3,3: 8

Matrix:
2 6 7
8 1 5
9 4 8
Left Diagonal Sum: 11
Right Diagonal Sum: 17

--------------------------------
Process exited after 12.97 seconds with return value 0
Press any key to continue . . .
```

<u>Task 2:</u>

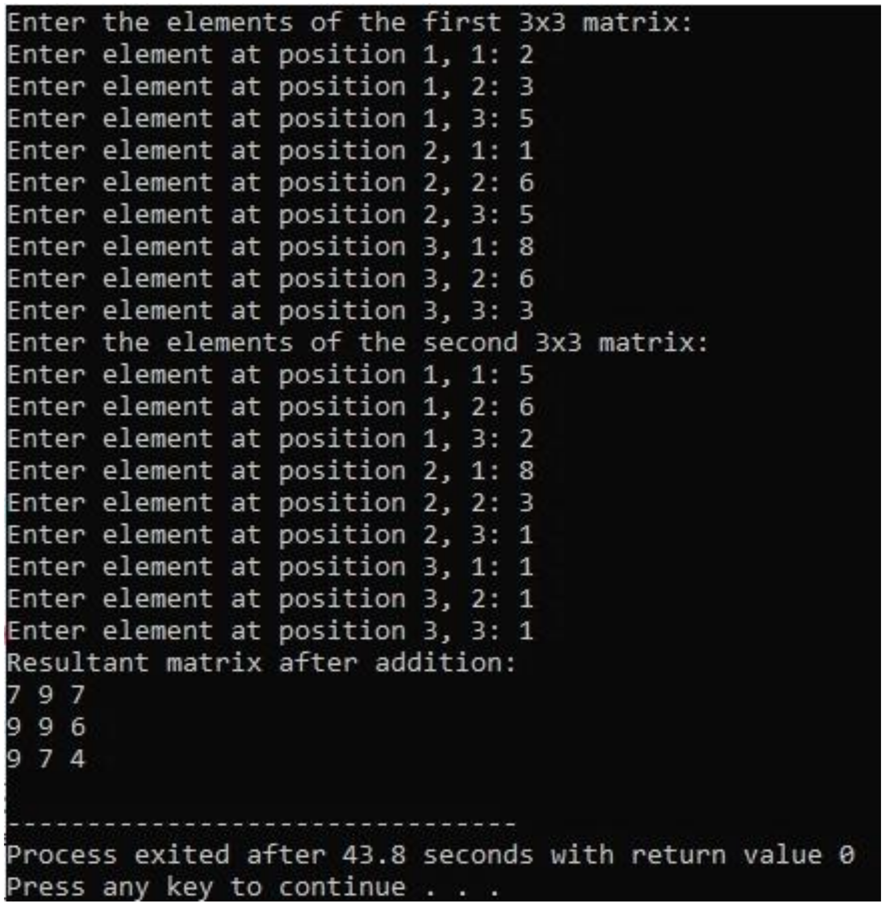
Write a function to add two 2D arrays of size 3x3.

Code:



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

int main() {
    int matrix1[size][size];
    int matrix2[size][size];
    int resultMatrix[size][size];
    // Input the elements of the first matrix
    cout << "Enter the elements of the first 3x3 matrix:" << endl;
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
            cout << "Enter element at position " << i + 1 << ", " << j + 1 << ": ";
            cin >> matrix1[i][j];
        }
    }
    // Input the elements of the second matrix
    cout << "Enter the elements of the second 3x3 matrix:" << endl;
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
            cout << "Enter element at position " << i + 1 << ", " << j + 1 << ": ";
            cin >> matrix2[i][j];
        }
    }
    // Add the matrices
    addMatrices(matrix1, matrix2, resultMatrix);
    // Print the result matrix
    cout << "Resultant matrix after addition:" << endl;
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
            cout << resultMatrix[i][j] << " ";
        }
        cout << endl;
    }

    return 0;
}
```

Output:

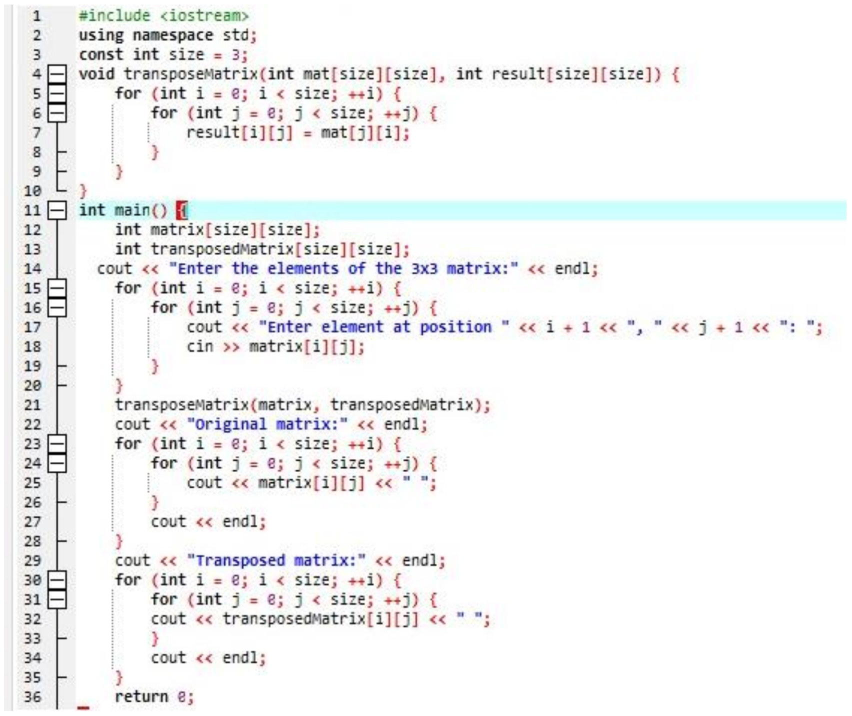


```
Enter the elements of the first 3x3 matrix:
Enter element at position 1, 1: 2
Enter element at position 1, 2: 3
Enter element at position 1, 3: 5
Enter element at position 2, 1: 1
Enter element at position 2, 2: 6
Enter element at position 2, 3: 5
Enter element at position 3, 1: 8
Enter element at position 3, 2: 6
Enter element at position 3, 3: 3
Enter the elements of the second 3x3 matrix:
Enter element at position 1, 1: 5
Enter element at position 1, 2: 6
Enter element at position 1, 3: 2
Enter element at position 2, 1: 8
Enter element at position 2, 2: 3
Enter element at position 2, 3: 1
Enter element at position 3, 1: 1
Enter element at position 3, 2: 1
Enter element at position 3, 3: 1
Resultant matrix after addition:
7 9 7
9 9 6
9 7 4

--------------------------------
Process exited after 43.8 seconds with return value 0
Press any key to continue . . .
```

Task 3:

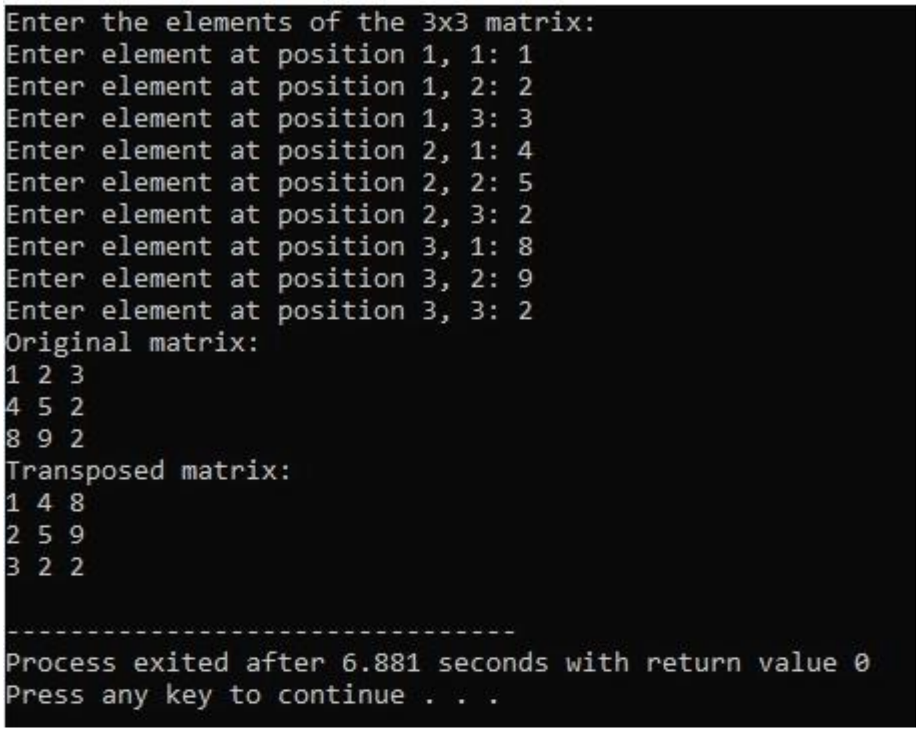
Using 2D arrays in C++, take transpose of a 3x3 matrix. Make a transpose function.

Code:



```cpp
#include <iostream>
using namespace std;
const int size = 3;
void transposeMatrix(int mat[size][size], int result[size][size]) {
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
            result[i][j] = mat[j][i];
        }
    }
}
int main() {
    int matrix[size][size];
    int transposedMatrix[size][size];
  cout << "Enter the elements of the 3x3 matrix:" << endl;
    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];
        }
    }
    transposeMatrix(matrix, transposedMatrix);
    cout << "Original matrix:" << endl;
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
            cout << matrix[i][j] << " ";
        }
        cout << endl;
    }
    cout << "Transposed matrix:" << endl;
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
        cout << transposedMatrix[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
```

Output:

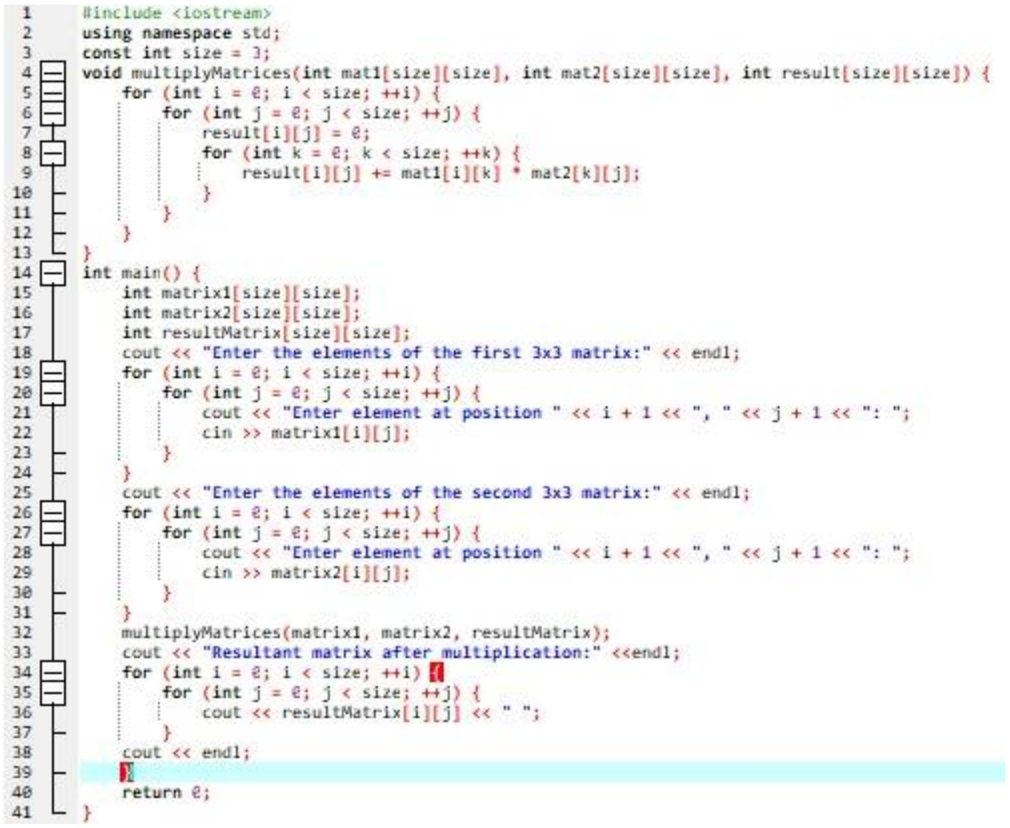


```
Enter the elements of the 3x3 matrix:
Enter element at position 1, 1: 1
Enter element at position 1, 2: 2
Enter element at position 1, 3: 3
Enter element at position 2, 1: 4
Enter element at position 2, 2: 5
Enter element at position 2, 3: 2
Enter element at position 3, 1: 8
Enter element at position 3, 2: 9
Enter element at position 3, 3: 2
Original matrix:
1 2 3
4 5 2
8 9 2
Transposed matrix:
1 4 8
2 5 9
3 2 2

--------------------------------
Process exited after 6.881 seconds with return value 0
Press any key to continue . . .
```

Task 4:

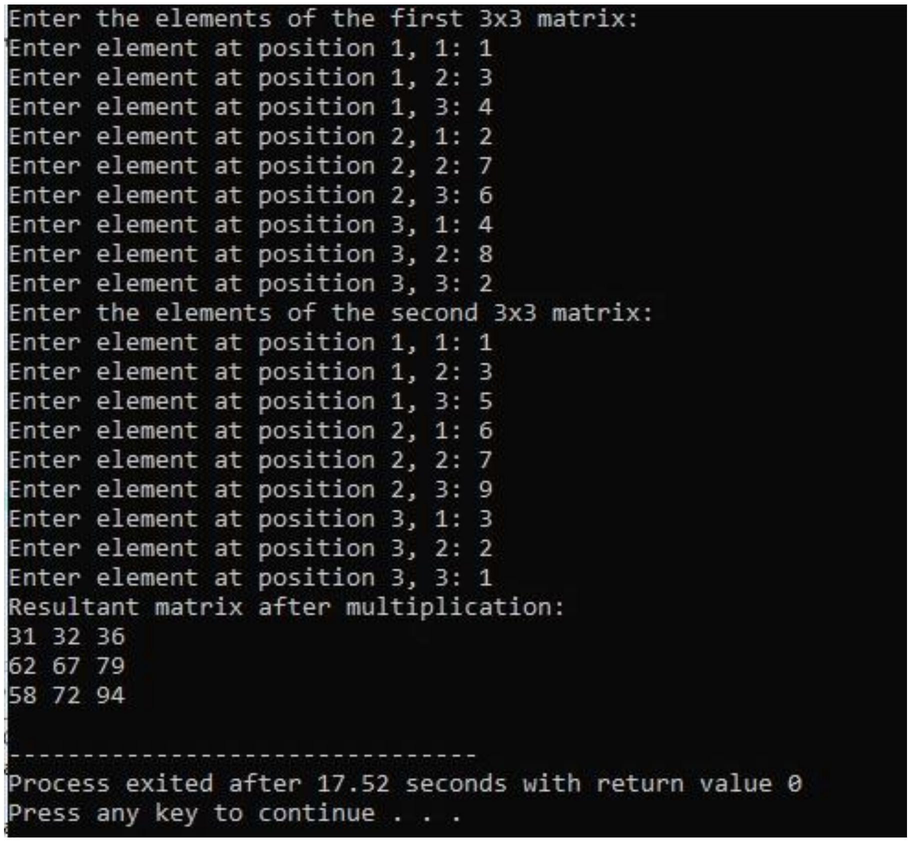
Using 2D arrays in C++, implement 3x3 matrix multiplication. Make a function.

Code:



```cpp
#include <iostream>
using namespace std;
const int size = 3;
void multiplyMatrices(int mat1[size][size], int mat2[size][size], int result[size][size]) {
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
            result[i][j] = 0;
            for (int k = 0; k < size; ++k) {
                result[i][j] += mat1[i][k] * mat2[k][j];
            }
        }
    }
}
int main() {
    int matrix1[size][size];
    int matrix2[size][size];
    int resultMatrix[size][size];
    cout << "Enter the elements of the first 3x3 matrix:" << endl;
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
            cout << "Enter element at position " << i + 1 << ", " << j + 1 << ": ";
            cin >> matrix1[i][j];
        }
    }
    cout << "Enter the elements of the second 3x3 matrix:" << endl;
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
            cout << "Enter element at position " << i + 1 << ", " << j + 1 << ": ";
            cin >> matrix2[i][j];
        }
    }
    multiplyMatrices(matrix1, matrix2, resultMatrix);
    cout << "Resultant matrix after multiplication:" <<endl;
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
            cout << resultMatrix[i][j] << " ";
        }
    cout << endl;
    }
    return 0;
}
```

Output:

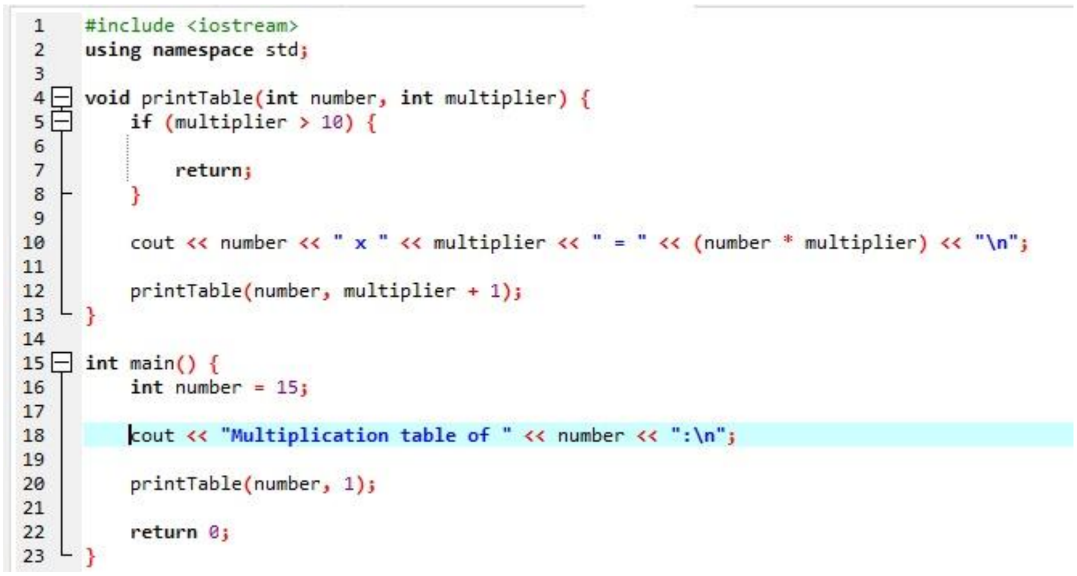


```
Enter the elements of the first 3x3 matrix:
Enter element at position 1, 1: 1
Enter element at position 1, 2: 3
Enter element at position 1, 3: 4
Enter element at position 2, 1: 2
Enter element at position 2, 2: 7
Enter element at position 2, 3: 6
Enter element at position 3, 1: 4
Enter element at position 3, 2: 8
Enter element at position 3, 3: 2
Enter the elements of the second 3x3 matrix:
Enter element at position 1, 1: 1
Enter element at position 1, 2: 3
Enter element at position 1, 3: 5
Enter element at position 2, 1: 6
Enter element at position 2, 2: 7
Enter element at position 2, 3: 9
Enter element at position 3, 1: 3
Enter element at position 3, 2: 2
Enter element at position 3, 3: 1
Resultant matrix after multiplication:
31 32 36
62 67 79
58 72 94

--------------------------------
Process exited after 17.52 seconds with return value 0
Press any key to continue . . .
```

Task 5:

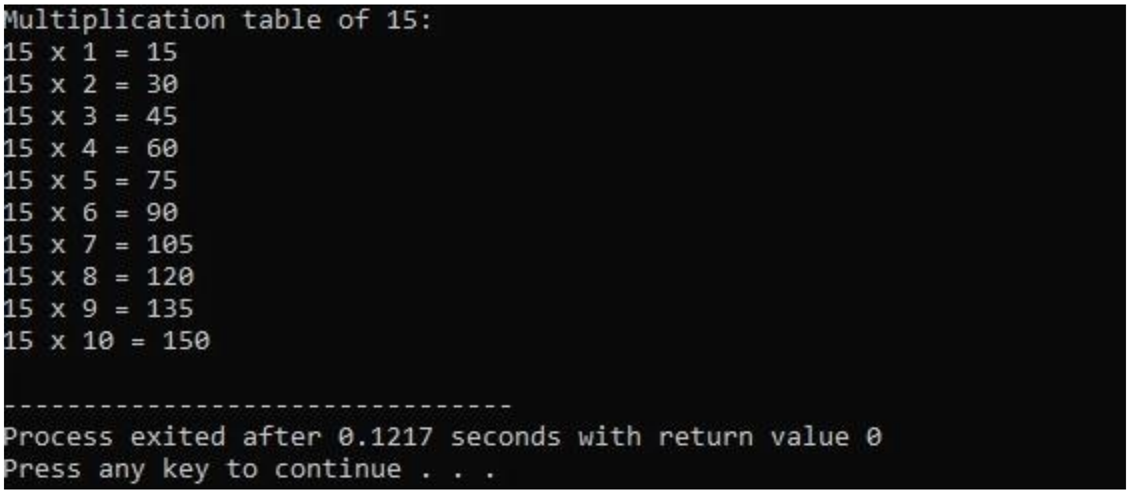
Print the multiplication table of 15 using recursion.

Code:



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

void printTable(int number, int multiplier) {
    if (multiplier > 10) {

        return;
    }

    cout << number << " x " << multiplier << " = " << (number * multiplier) << "\n";

    printTable(number, multiplier + 1);
}

int main() {
    int number = 15;

    cout << "Multiplication table of " << number << ":\n";

    printTable(number, 1);

    return 0;
}
```

Output:

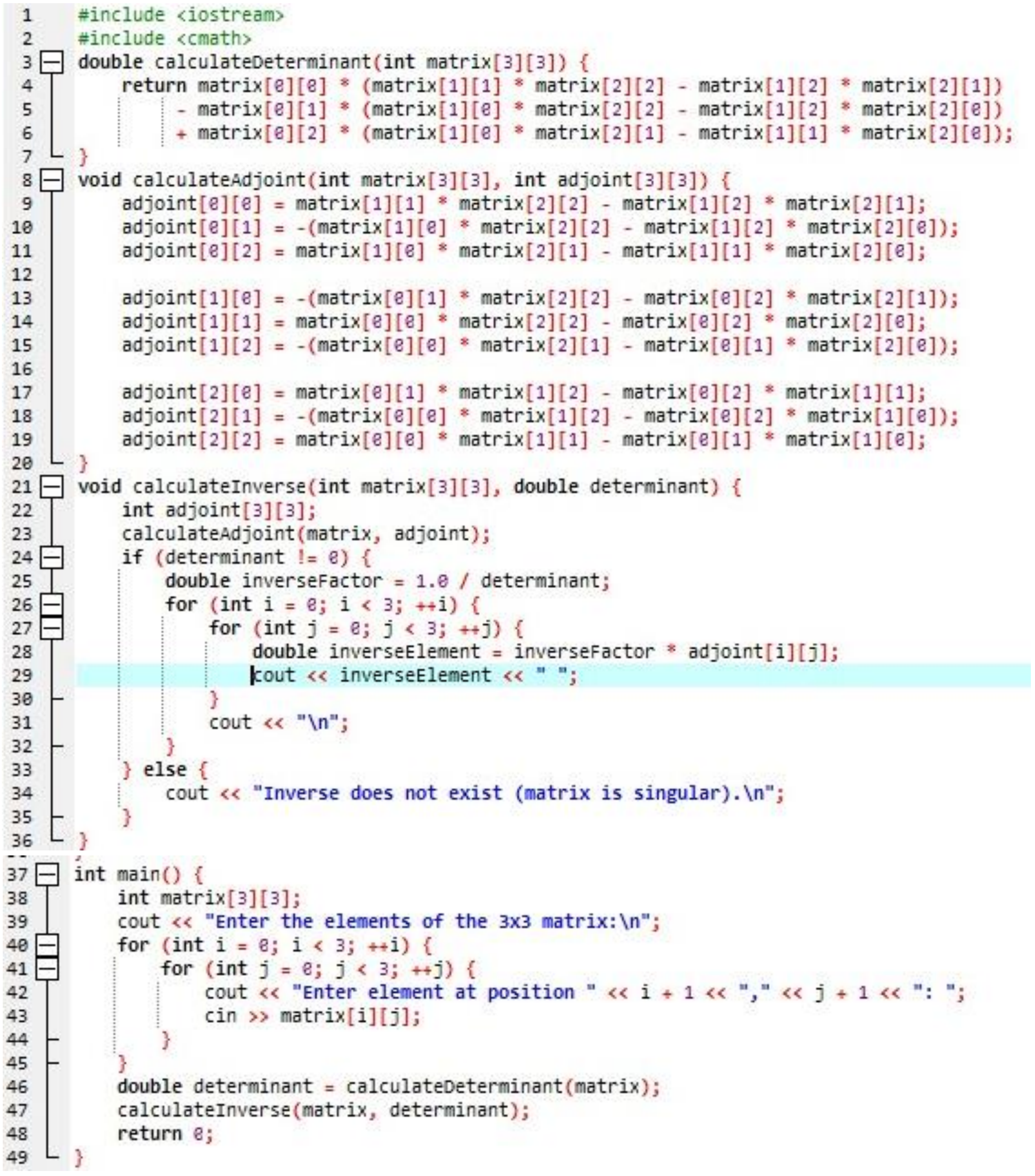


```
Multiplication table of 15:
15 x 1 = 15
15 x 2 = 30
15 x 3 = 45
15 x 4 = 60
15 x 5 = 75
15 x 6 = 90
15 x 7 = 105
15 x 8 = 120
15 x 9 = 135
15 x 10 = 150

--------------------------------
Process exited after 0.1217 seconds with return value 0
Press any key to continue . . .
```

Home Task:

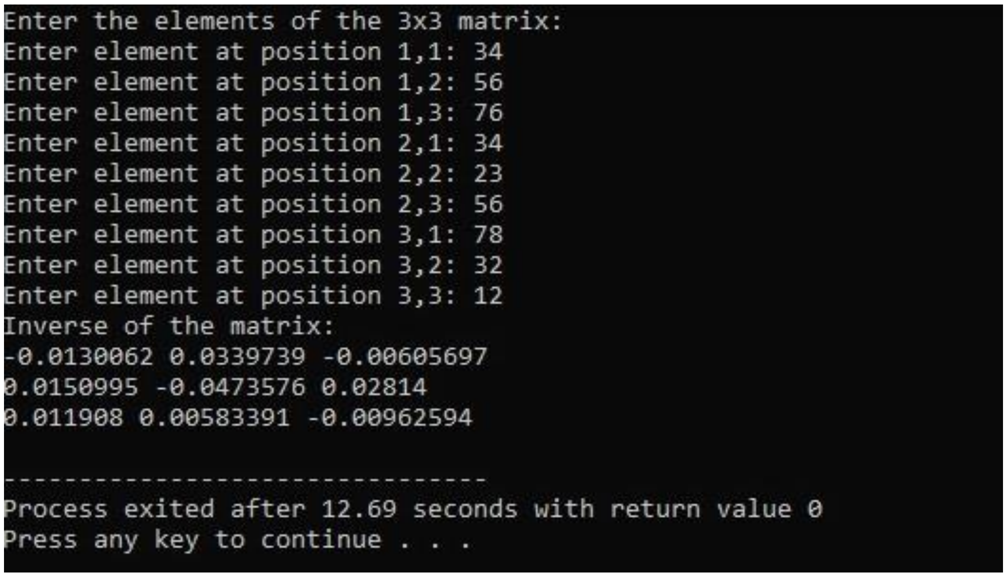
Write a C++ program to take the inverse of a 3x3 matrix using its determinant and adjoint.

Code:



```cpp
#include <iostream>
#include <cmath>
double calculateDeterminant(int matrix[3][3]) {
    return matrix[0][0] * (matrix[1][1] * matrix[2][2] - matrix[1][2] * matrix[2][1])
         - matrix[0][1] * (matrix[1][0] * matrix[2][2] - matrix[1][2] * matrix[2][0])
         + matrix[0][2] * (matrix[1][0] * matrix[2][1] - matrix[1][1] * matrix[2][0]);
}
void calculateAdjoint(int matrix[3][3], int adjoint[3][3]) {
    adjoint[0][0] = matrix[1][1] * matrix[2][2] - matrix[1][2] * matrix[2][1];
    adjoint[0][1] = -(matrix[1][0] * matrix[2][2] - matrix[1][2] * matrix[2][0]);
    adjoint[0][2] = matrix[1][0] * matrix[2][1] - matrix[1][1] * matrix[2][0];

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

    adjoint[2][0] = matrix[0][1] * matrix[1][2] - matrix[0][2] * matrix[1][1];
    adjoint[2][1] = -(matrix[0][0] * matrix[1][2] - matrix[0][2] * matrix[1][0]);
    adjoint[2][2] = matrix[0][0] * matrix[1][1] - matrix[0][1] * matrix[1][0];
}
void calculateInverse(int matrix[3][3], double determinant) {
    int adjoint[3][3];
    calculateAdjoint(matrix, adjoint);
    if (determinant != 0) {
        double inverseFactor = 1.0 / determinant;
        for (int i = 0; i < 3; ++i) {
            for (int j = 0; j < 3; ++j) {
                double inverseElement = inverseFactor * adjoint[i][j];
                cout << inverseElement << " ";
            }
            cout << "\n";
        }
    } else {
        cout << "Inverse does not exist (matrix is singular).\n";
    }
}
int main() {
    int matrix[3][3];
    cout << "Enter the elements of the 3x3 matrix:\n";
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << "Enter element at position " << i + 1 << "," << j + 1 << ": ";
            cin >> matrix[i][j];
        }
    }
    double determinant = calculateDeterminant(matrix);
    calculateInverse(matrix, determinant);
    return 0;
}
```

Output:



```
Enter the elements of the 3x3 matrix:
Enter element at position 1,1: 34
Enter element at position 1,2: 56
Enter element at position 1,3: 76
Enter element at position 2,1: 34
Enter element at position 2,2: 23
Enter element at position 2,3: 56
Enter element at position 3,1: 78
Enter element at position 3,2: 32
Enter element at position 3,3: 12
Inverse of the matrix:
-0.0130062 0.0339739 -0.00605697
0.0150995 -0.0473576 0.02814
0.011908 0.00583391 -0.00962594


--------------------------------
Process exited after 12.69 seconds with return value 0
Press any key to continue . . .
```

Conclusion:

We learned about multi-dimensional arrays, especially 2D and 3D arrays. We learned how to initialize and declare multidimensional arrays. We also learned how to declare and form functions and how to call them inside the main function.