Name Muhammad Asim Shah
Cms 470574

Lab Manual 09

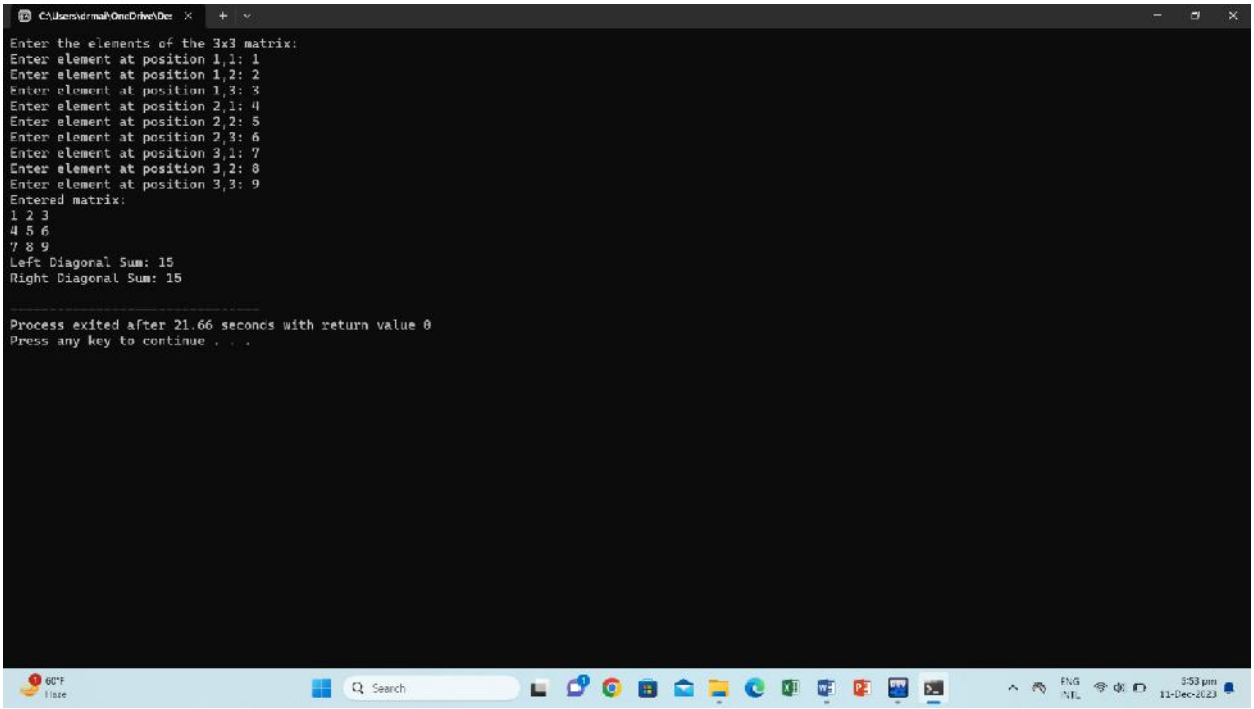
Make 2D Array in C++ and print left diagonal and right diagonal sum of a 3x3 matrix

CODE:

```
#include <iostream>
using namespace std;
int main() {
   int size = 3;
   int matrix[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];
      }
   }
   cout << "Entered matrix:" << endl;
   for (int i = 0; i < size; i++) {
      for (int j = 0; j < size; j++) {
         cout << matrix[i][j] << " ";
      }
      cout << endl;
   }
   int leftDiagonalSum = 0;
   for (int i = 0; i < size; i++) {
```

```
        leftDiagonalSum += matrix[i][i];
    }
    int rightDiagonalSum = 0;
    for (int i = 0; i < size; i++) {
        rightDiagonalSum += matrix[i][size - 1 - i];
    }
    cout << "Left Diagonal Sum: " << leftDiagonalSum << endl;
    cout << "Right Diagonal Sum: " << rightDiagonalSum << endl;
    return 0;
}
```

OUTPUT:



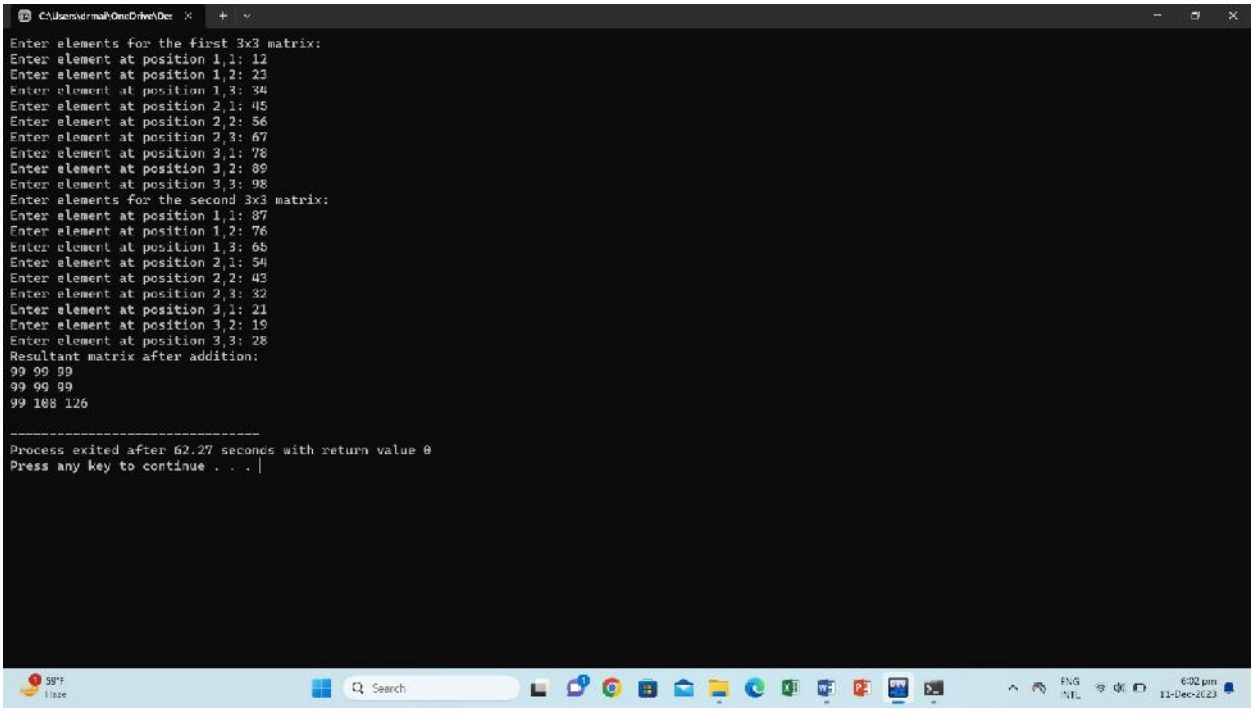
Write a function to add two 2D arrays of size 3x3.

CODE:

```
#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], matrix2[size][size], resultMatrix[size][size];
    cout << "Enter elements for 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 elements for 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];
        }
    }
    addMatrices(matrix1, matrix2, resultMatrix);
    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:



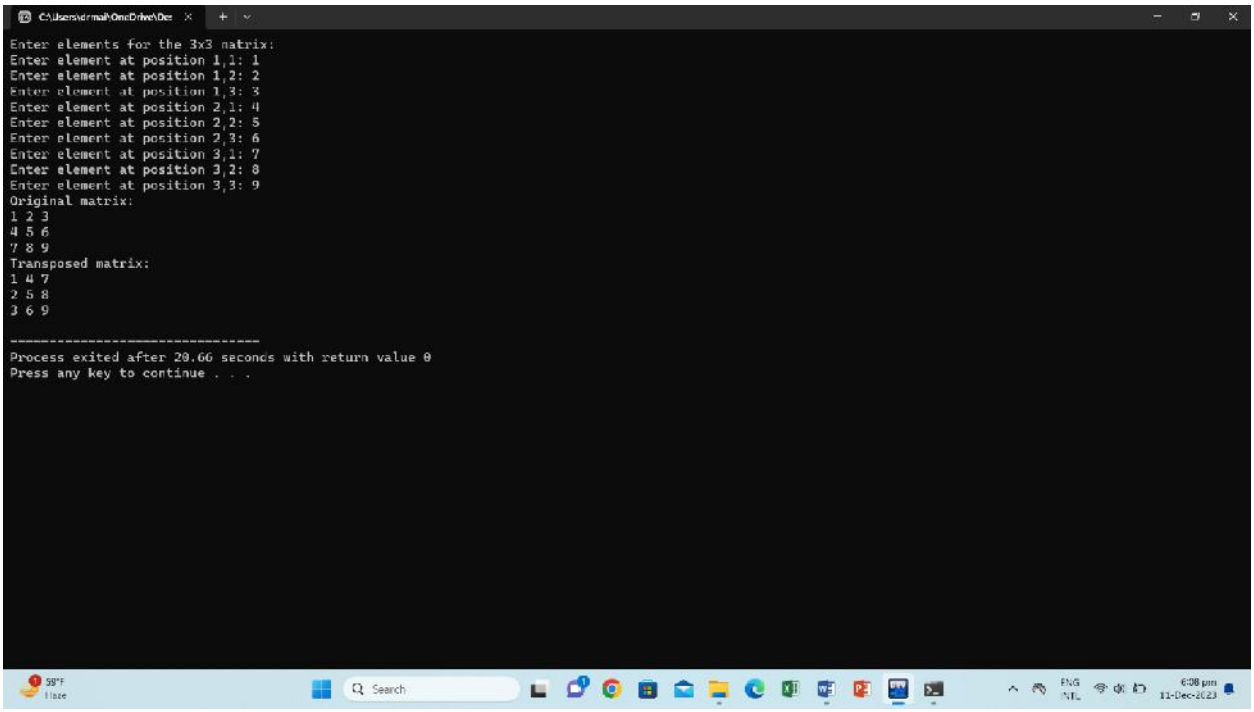
Using 2D arrays in C++, take transpose of a 3x3 matrix. Make a transpose function

CODE:

```
#include <iostream>
using namespace std;
const int size = 3;
void transposeMatrix(int original[size][size], int transposed[size][size]) {
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
            transposed[j][i] = original[i][j];
        }
    }
}
int main() {
    int matrix[size][size], transposedMatrix[size][size];
    cout << "Enter elements for 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:



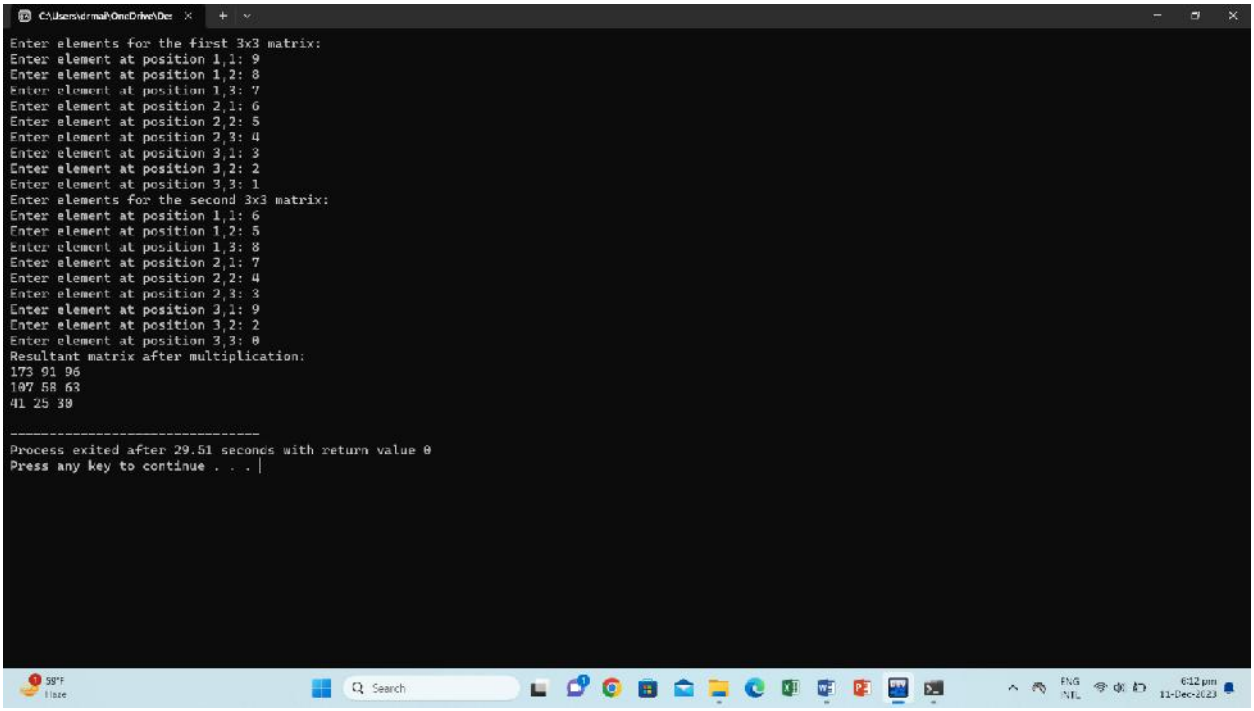
Using 2D arrays in C++, implement 3x3 matrix multiplication. Make a function.

CODE:

```
#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], matrix2[size][size], resultMatrix[size][size];
   cout << "Enter elements for 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 elements for 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:

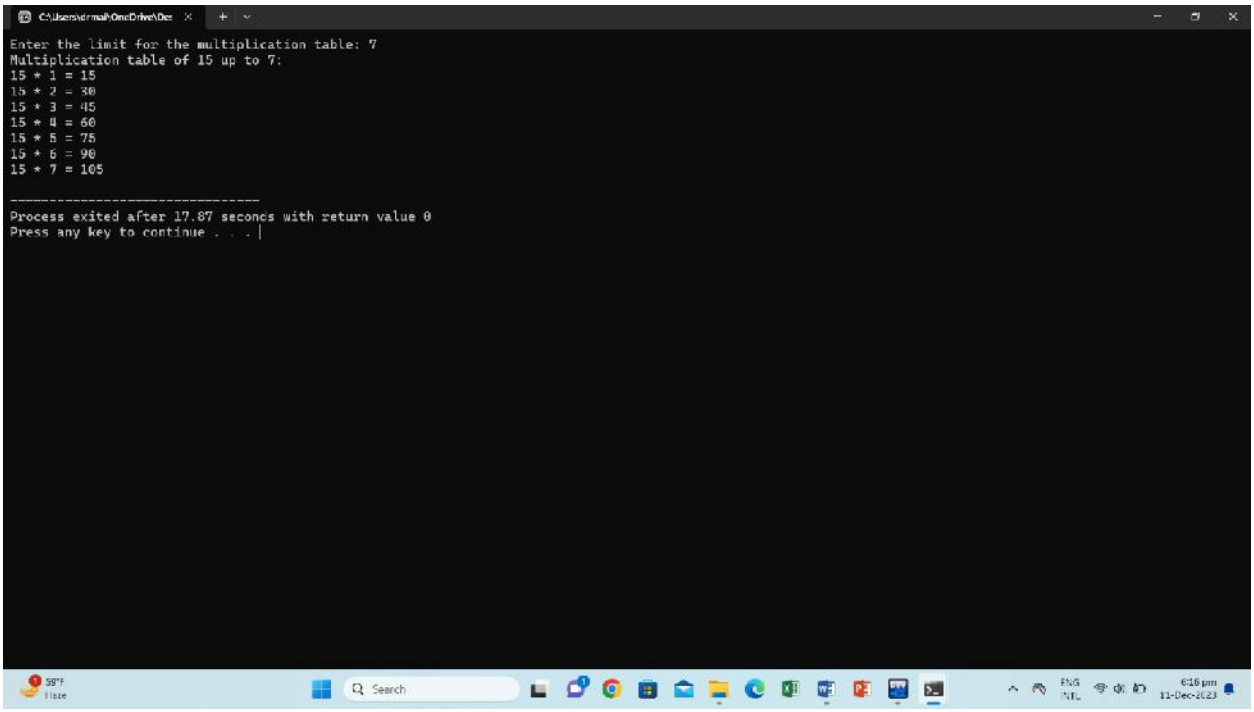


Print the multiplication table of 15 using recursion

CODE:

```
#include <iostream>
using namespace std;
void printMultiplicationTable(int number, int limit, int current) {
   if (current > limit) {
      return;
   }
   cout << number << " * " << current << " = " << number * current << endl;
   printMultiplicationTable(number, limit, current + 1);
}
int main() {
   const int tableOf = 15;
   int limit;
   cout << "Enter the limit for the multiplication table: ";
   cin >> limit;
   cout << "Multiplication table of " << tableOf << " up to " << limit << ":" << endl;
   printMultiplicationTable(tableOf, limit, 1);
   return 0;
}
```

OUTPUT:



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

CODE:

```
#include <iostream>

using namespace std;

const int size = 3;

int determinant(int mat[size][size]) {
   return mat[0][0] * (mat[1][1] * mat[2][2] - mat[2][1] * mat[1][2]) -
          mat[0][1] * (mat[1][0] * mat[2][2] - mat[2][0] * mat[1][2]) +
          mat[0][2] * (mat[1][0] * mat[2][1] - mat[2][0] * mat[1][1]);}

void transposeMatrix(int original[size][size], int transposed[size][size]) {
   for (int i = 0; i < size; ++i) {
      for (int j = 0; j < size; ++j) {
```

```
        transposed[j][i] = original[i][j];}  }  }
void cofactorMatrix(int mat[size][size], int cofactor[size][size]) {
   for (int i = 0; i < size; ++i) {
      for (int j = 0; j < size; ++j) {
         int sign = ((i + j) % 2 == 0) ? 1 : -1;
         cofactor[i][j] = sign * determinant(mat[i][j]);}  }  }
void inverseMatrix(int mat[size][size], int inverse[size][size]) {
   int det = determinant(mat);
   if (det == 0) {
      cout << "The matrix is singular, and its inverse does not exist." << endl;
      return 0;}
   int cofactorMatrix[size][size];
   cofactorMatrix(mat, cofactorMatrix);
   int adjoint[size][size];
   transposeMatrix(cofactorMatrix, adjoint);
   for (int i = 0; i < size; ++i) {
      for (int j = 0; j < size; ++j) {
         inverse[i][j] = adjoint[i][j] / det;}   }  }
void printMatrix(int mat[size][size]) {
   for (int i = 0; i < size; ++i) {
      for (int j = 0; j < size; ++j) {
         cout << mat[i][j] << " ";}
      cout << endl;}
int main() {
```

```
int matrix[size][size], inverseMatrixResult[size][size];
cout << "Enter elements for 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];}    }
inverseMatrix(matrix, inverseMatrixResult);
cout << "Original matrix:" << endl;
printMatrix(matrix);
cout << "\nInverse matrix:" << endl;
printMatrix(inverseMatrixResult);
return 0;}
```

OUTPUT: