NATIONAL UNIVERSITY OF SCIENCES & TECHNOLOGY

SCHOOL OF MECHANICAL AND MANUFACTURING ENGINEERING

SEMESTER # 01

CLASS: - ME 15 [SEC A]

KASHIF NADEEM KAYANI

456466

Fundamentals of Programming

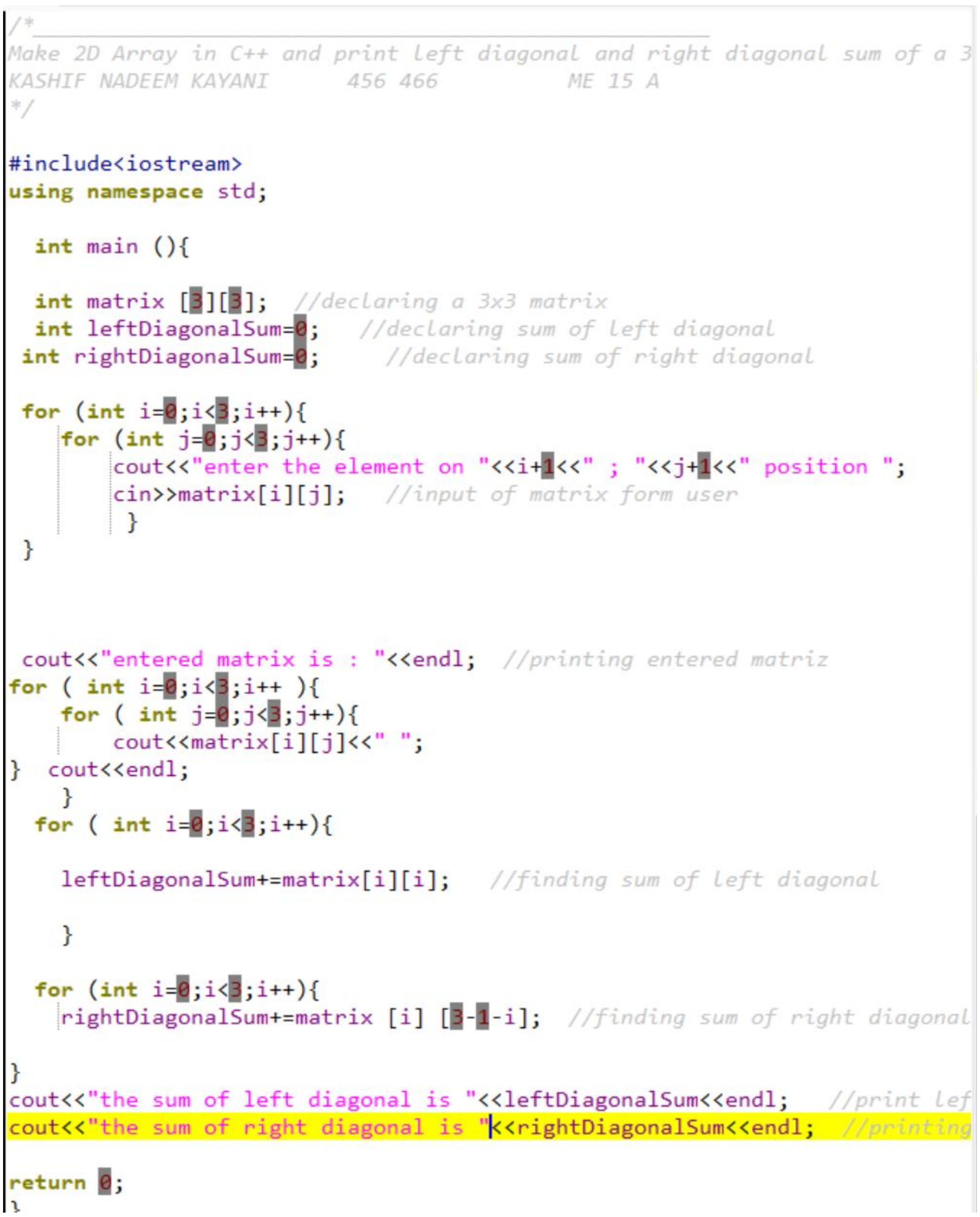
LAB MANUAL 09

Date of Submission 12 DEC 2023

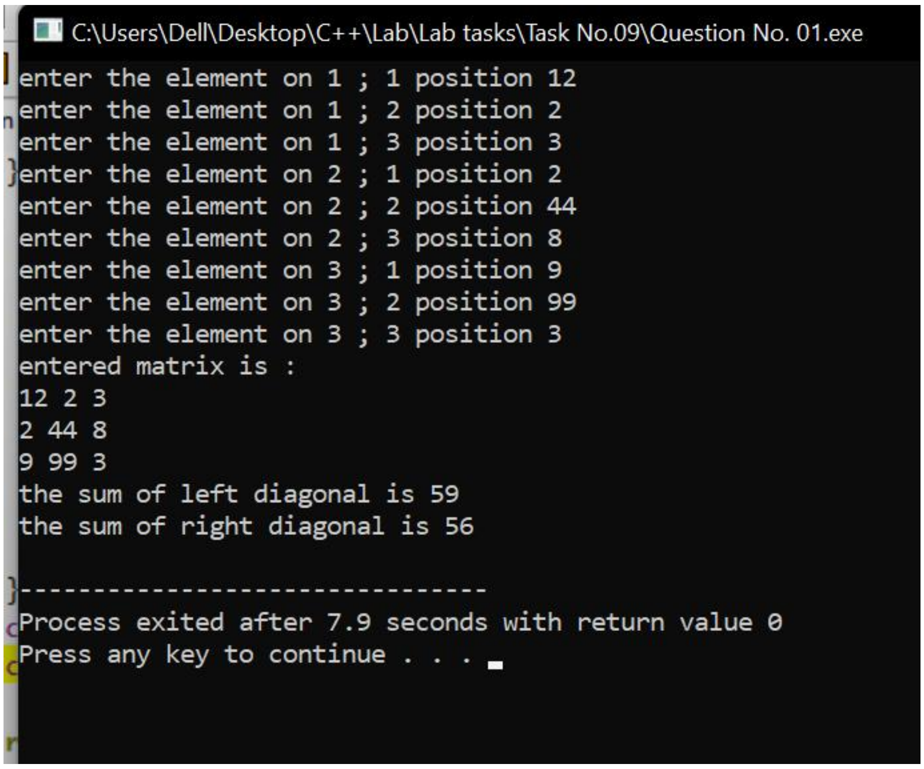
Submitted to MUHAMMAD AFFAN

QUESTION NUMBER 01

```
/*____________________________________________________
Make 2D Array in C++ and print left diagonal and right diagonal sum of a 3x3 matrix
KASHIF NADEEM KAYANI          456 466                  ME 15 A
*/

#include<iostream>
using namespace std;

   int main (){

   int matrix [3][3];    //declaring a 3x3 matrix
   int leftDiagonalSum=0;      //declaring sum of left diagonal
  int rightDiagonalSum=0;          //declaring sum of right diagonal

  for (int i=0;i<3;i++){
            for (int j=0;j<3;j++){
                       cout<<"enter the element on "<<i+1<<" ; "<<j+1<<" position ";
                       cin>>matrix[i][j];      //input of matrix form user
             }
  }



  cout<<"entered matrix is : "<<endl;    //printing entered matriz
for ( int i=0;i<3;i++ ){
            for ( int j=0;j<3;j++){
                       cout<<matrix[i][j]<<" ";
}    cout<<endl;
            }
   for ( int i=0;i<3;i++){

            leftDiagonalSum+=matrix[i][i];      //finding sum of left diagonal

      }

   for (int i=0;i<3;i++){
            rightDiagonalSum+=matrix [i] [3-1-i];    //finding sum of right diagonal

}
cout<<"the sum of left diagonal is "<<leftDiagonalSum<<endl;      //print left diagonal sum
cout<<"the sum of right diagonal is "<<rightDiagonalSum<<endl;    //printing right diagonal sum

return 0;
}
```



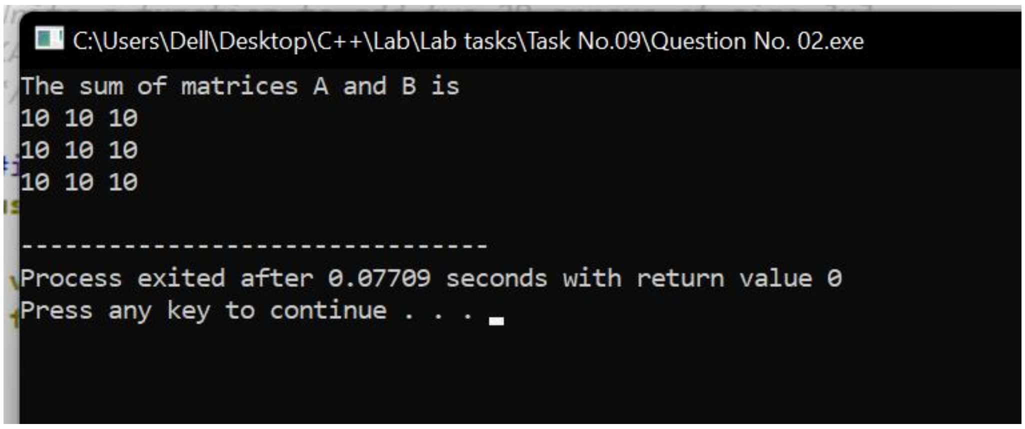
```
/*_____________________________________________________
Make 2D Array in C++ and print left diagonal and right diagonal sum of a 3
KASHIF NADEEM KAYANI      456 466          ME 15 A
*/

#include<iostream>
using namespace std;

  int main (){

  int matrix [3][3];  //declaring a 3x3 matrix
  int leftDiagonalSum=0;   //declaring sum of left diagonal
 int rightDiagonalSum=0;     //declaring sum of right diagonal

 for (int i=0;i<3;i++){
    for (int j=0;j<3;j++){
        cout<<"enter the element on "<<i+1<<" ; "<<j+1<<" position ";
        cin>>matrix[i][j];    //input of matrix form user
         }
 }



 cout<<"entered matrix is : "<<endl;  //printing entered matriz
for ( int i=0;i<3;i++ ){
    for ( int j=0;j<3;j++){
        cout<<matrix[i][j]<<" ";
}  cout<<endl;
    }
  for ( int i=0;i<3;i++){

    leftDiagonalSum+=matrix[i][i];    //finding sum of left diagonal

    }

  for (int i=0;i<3;i++){
    rightDiagonalSum+=matrix [i] [3-1-i];  //finding sum of right diagonal

}
cout<<"the sum of left diagonal is "<<leftDiagonalSum<<endl;   //print lef
cout<<"the sum of right diagonal is "<<rightDiagonalSum<<endl;  //printing

return 0;
}
```


C:\Users\Dell\Desktop\C++\Lab\Lab tasks\Task No.09\Question No. 01.exe

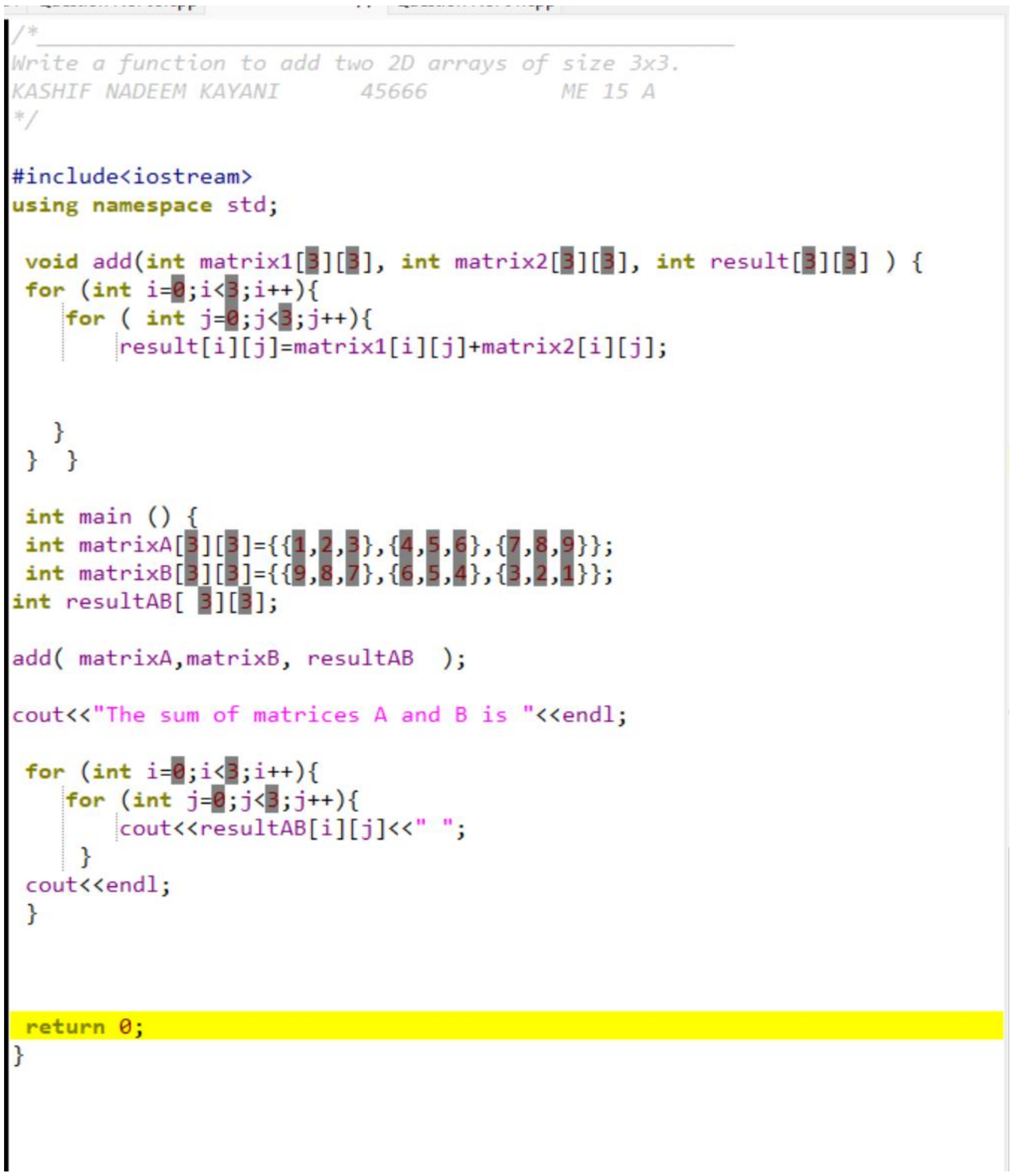
```
enter the element on 1 ; 1 position 12
enter the element on 1 ; 2 position 2
enter the element on 1 ; 3 position 3
enter the element on 2 ; 1 position 2
enter the element on 2 ; 2 position 44
enter the element on 2 ; 3 position 8
enter the element on 3 ; 1 position 9
enter the element on 3 ; 2 position 99
enter the element on 3 ; 3 position 3
entered matrix is :
12 2 3
2 44 8
9 99 3
the sum of left diagonal is 59
the sum of right diagonal is 56

--------------------------------
Process exited after 7.9 seconds with return value 0
Press any key to continue . . .
```

QUESTION NUMBER 02

```
/*_______________________________________________________
Write a function to add two 2D arrays of size 3x3.
KASHIF NADEEM KAYANI          45666                ME 15 A
*/

#include<iostream>
using namespace std;

  void add(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];



      }
  }   }

  int main () {
  int matrixA[3][3]={{1,2,3},{4,5,6},{7,8,9}};
  int matrixB[3][3]={{9,8,7},{6,5,4},{3,2,1}};
int resultAB[ 3][3];

add( matrixA,matrixB, resultAB    );

cout<<"The sum of matrices A and B is "<<endl;

  for (int i=0;i<3;i++){
            for (int j=0;j<3;j++){
                      cout<<resultAB[i][j]<<" ";
             }
  cout<<endl;
  }



  return 0;
}
```



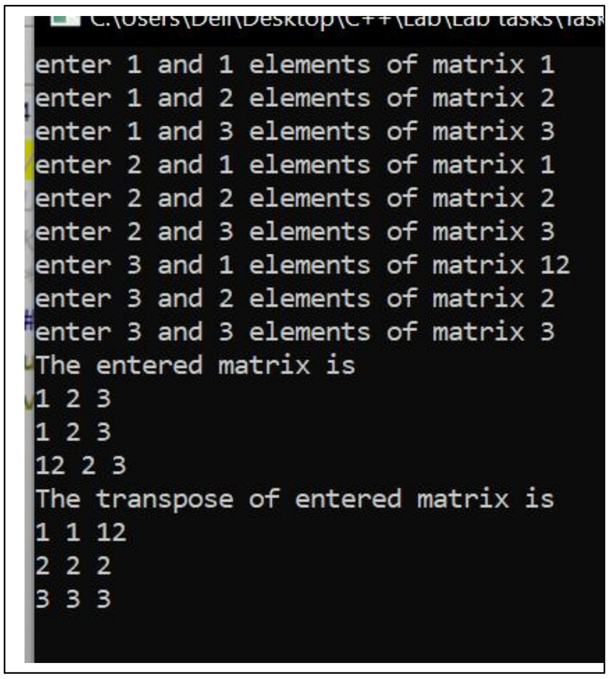
```
C:\Users\Dell\Desktop\C++\Lab\Lab tasks\Task No.09\Question No. 02.exe
The sum of matrices A and B is
10 10 10
10 10 10
10 10 10

--------------------------------
Process exited after 0.07709 seconds with return value 0
Press any key to continue . . .
```



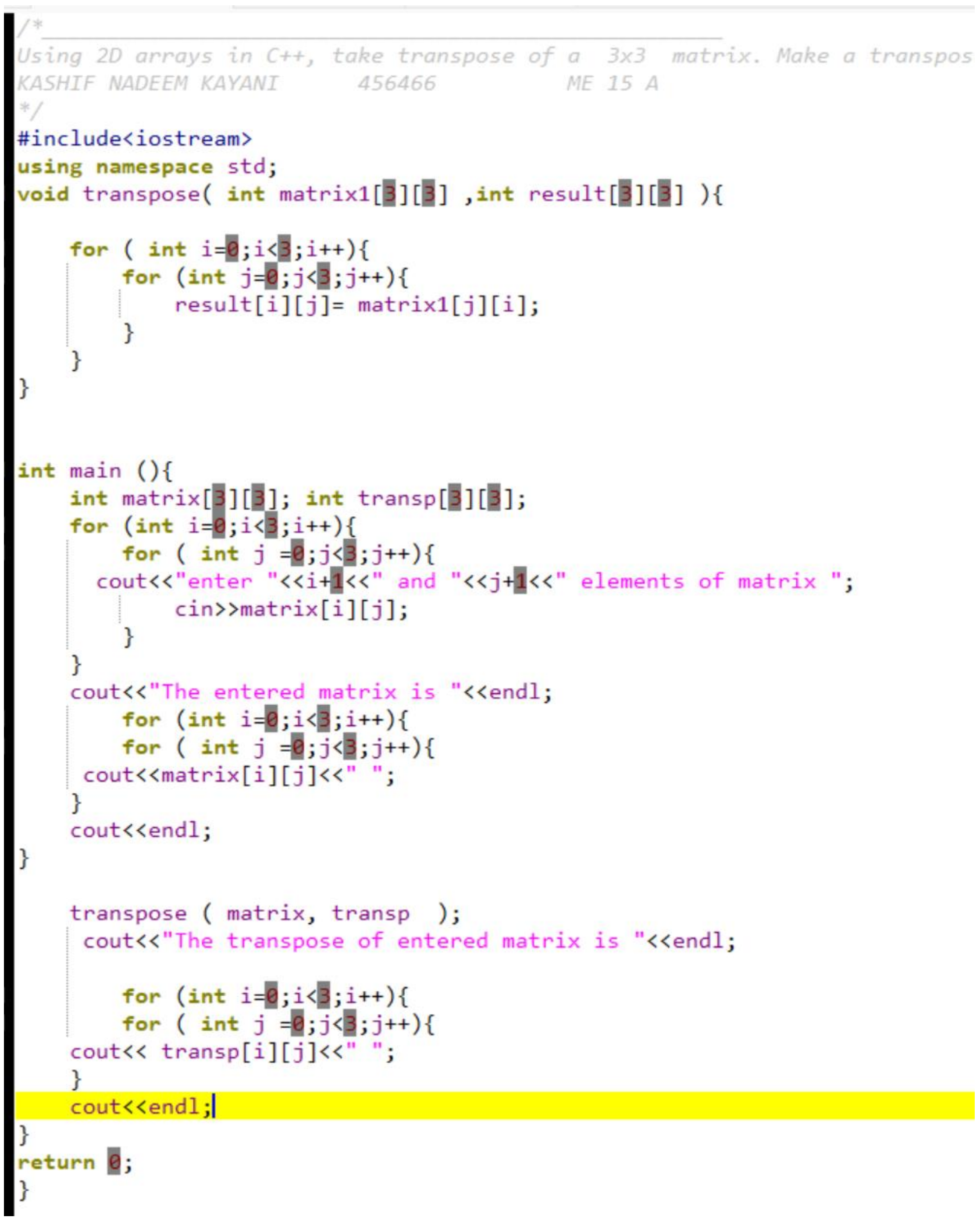
```cpp
/*______________________________________________________________
Write a function to add two 2D arrays of size 3x3.
KASHIF NADEEM KAYANI      45666          ME 15 A
*/

#include<iostream>
using namespace std;

 void add(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];


   }
 }  }

 int main () {
 int matrixA[3][3]={{1,2,3},{4,5,6},{7,8,9}};
 int matrixB[3][3]={{9,8,7},{6,5,4},{3,2,1}};
int resultAB[ 3][3];

add( matrixA,matrixB, resultAB  );

cout<<"The sum of matrices A and B is "<<endl;

 for (int i=0;i<3;i++){
    for (int j=0;j<3;j++){
        cout<<resultAB[i][j]<<" ";
     }
 cout<<endl;
 }



 return 0;
}
```

QUESTION NUMBER 03

```
/*_______________________________________________________
Using 2D arrays in C++, take transpose of a    3x3    matrix. Make a transpose function
KASHIF NADEEM KAYANI           456466                  ME 15 A
*/
#include<iostream>
using namespace std;
void transpose( int matrix1[3][3] ,int result[3][3] ){

        for ( int i=0;i<3;i++){
                for (int j=0;j<3;j++){
                        result[i][j]= matrix1[j][i];
                }
        }
}
int main (){
        int matrix[3][3]; int transp[3][3];
        for (int i=0;i<3;i++){
                for ( int j =0;j<3;j++){
          cout<<"enter "<<i+1<<" and "<<j+1<<" elements of matrix ";
                        cin>>matrix[i][j];
                }
        }
        cout<<"The entered matrix is "<<endl;
                for (int i=0;i<3;i++){
                for ( int j =0;j<3;j++){
         cout<<matrix[i][j]<<" ";
    }
    cout<<endl;
}

        transpose ( matrix, transp    );
         cout<<"The transpose of entered matrix is "<<endl;

                for (int i=0;i<3;i++){
                for ( int j =0;j<3;j++){
        cout<< transp[i][j]<<" ";
    }
    cout<<endl;
}
return 0;
}
```



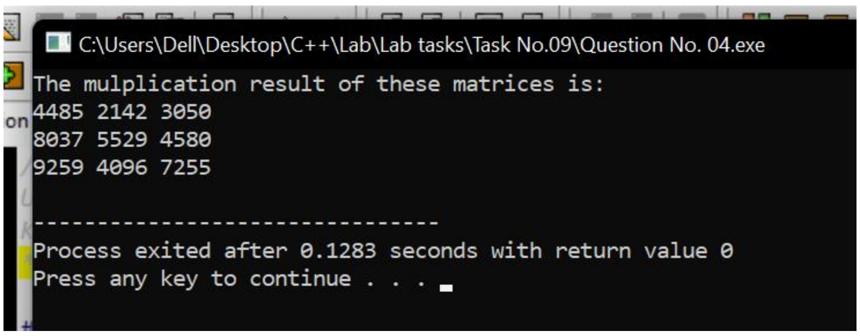
```
C:\Users\Dell\Desktop\C++\Lab\Lab tasks\Tas
enter 1 and 1 elements of matrix 1
enter 1 and 2 elements of matrix 2
enter 1 and 3 elements of matrix 3
enter 2 and 1 elements of matrix 1
enter 2 and 2 elements of matrix 2
enter 2 and 3 elements of matrix 3
enter 3 and 1 elements of matrix 12
enter 3 and 2 elements of matrix 2
enter 3 and 3 elements of matrix 3
The entered matrix is
1 2 3
1 2 3
12 2 3
The transpose of entered matrix is
1 1 12
2 2 2
3 3 3
```



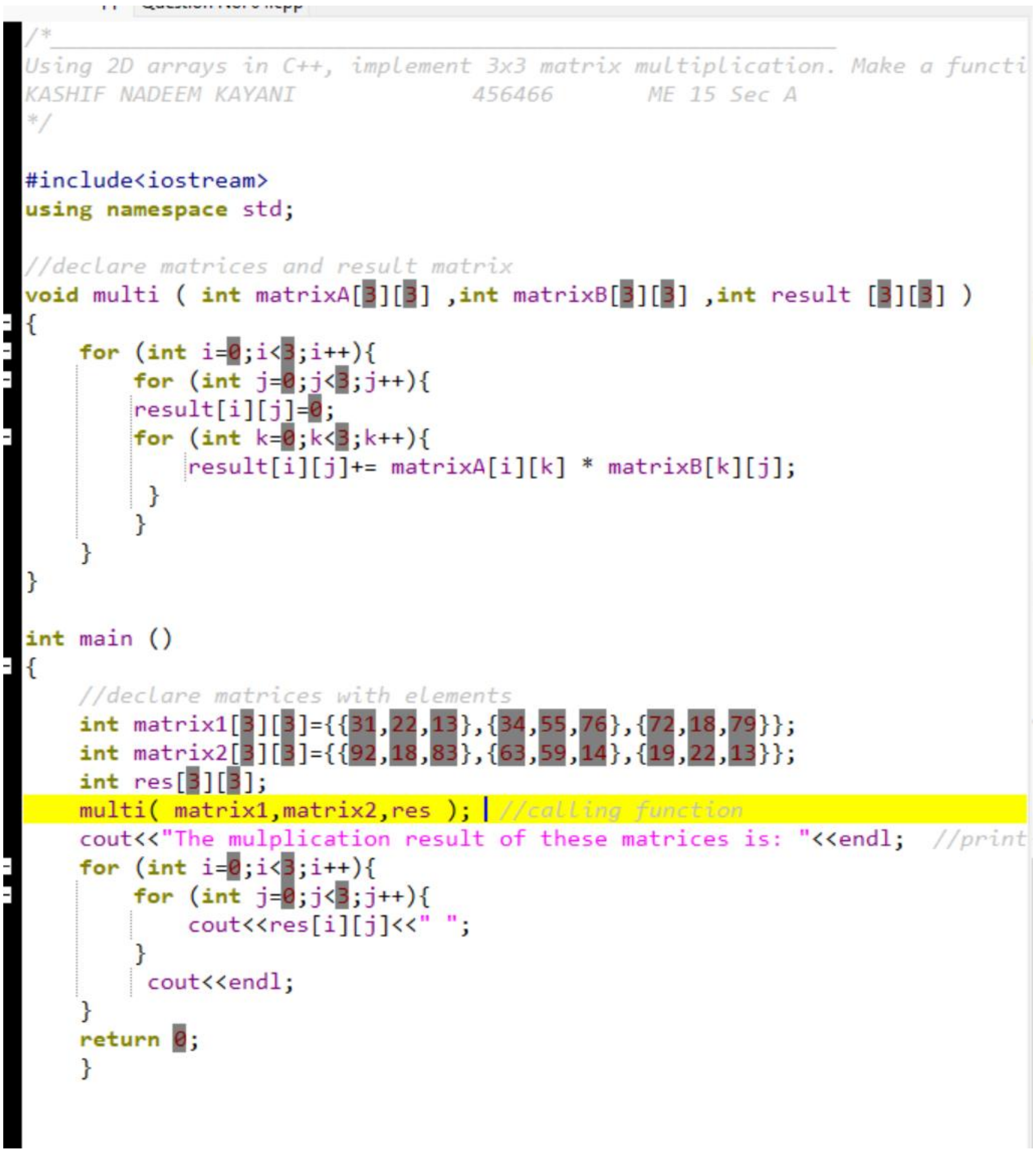
```
/*___________________________________________________________
Using 2D arrays in C++, take transpose of a  3x3  matrix. Make a transpos
KASHIF NADEEM KAYANI      456466          ME 15 A
*/
#include<iostream>
using namespace std;
void transpose( int matrix1[3][3] ,int result[3][3] ){

    for ( int i=0;i<3;i++){
        for (int j=0;j<3;j++){
            result[i][j]= matrix1[j][i];
        }
    }
}


int main (){
    int matrix[3][3]; int transp[3][3];
    for (int i=0;i<3;i++){
        for ( int j =0;j<3;j++){
      cout<<"enter "<<i+1<<" and "<<j+1<<" elements of matrix ";
            cin>>matrix[i][j];
        }
    }
    cout<<"The entered matrix is "<<endl;
        for (int i=0;i<3;i++){
        for ( int j =0;j<3;j++){
     cout<<matrix[i][j]<<" ";
    }
    cout<<endl;
}

    transpose ( matrix, transp  );
     cout<<"The transpose of entered matrix is "<<endl;

        for (int i=0;i<3;i++){
        for ( int j =0;j<3;j++){
    cout<< transp[i][j]<<" ";
    }
    cout<<endl;
}
return 0;
}
```

QUESTION NUMBER 04

```
/*_______________________________________________________________
Using 2D arrays in C++, implement 3x3 matrix multiplication. Make a function.
KASHIF NADEEM KAYANI                    456466            ME 15 Sec A
*/

#include<iostream>
using namespace std;

//declare matrices and result matrix
void multi ( int matrixA[3][3] ,int matrixB[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]+= matrixA[i][k] * matrixB[k][j];
                  }
                }
        }
}

int main ()
{
        //declare matrices with elements
        int matrix1[3][3]={{31,22,13},{34,55,76},{72,18,79}};
        int matrix2[3][3]={{92,18,83},{63,59,14},{19,22,13}};
        int res[3][3];
        multi( matrix1,matrix2,res );    //calling function
        cout<<"The mulplication result of these matrices is: "<<endl;    //printing results
        for (int i=0;i<3;i++){
                for (int j=0;j<3;j++){
                        cout<<res[i][j]<<" ";
                }
                  cout<<endl;
        }
        return 0;
        }
```



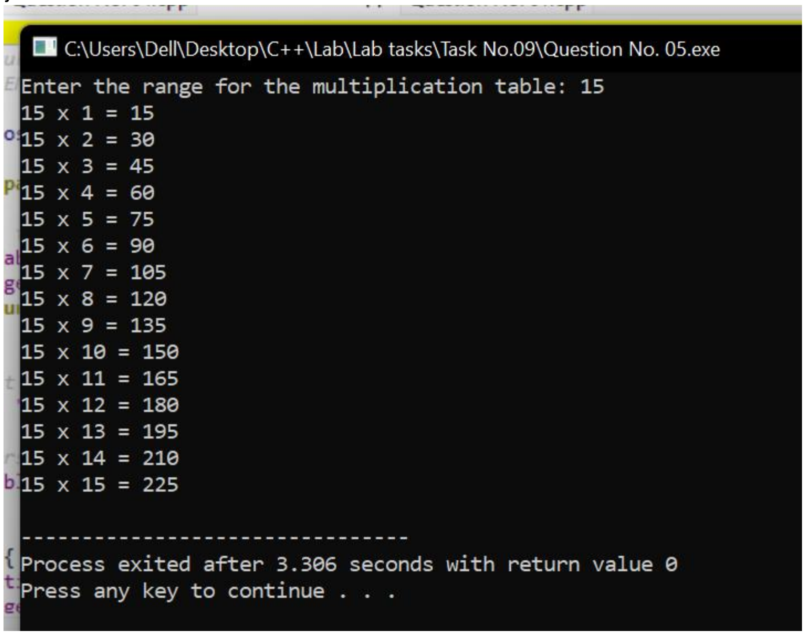
```
C:\Users\Dell\Desktop\C++\Lab\Lab tasks\Task No.09\Question No. 04.exe
The mulplication result of these matrices is:
4485 2142 3050
8037 5529 4580
9259 4096 7255

--------------------------------
Process exited after 0.1283 seconds with return value 0
Press any key to continue . . .
```



```
/*_______________________________________________________________
Using 2D arrays in C++, implement 3x3 matrix multiplication. Make a functi
KASHIF NADEEM KAYANI            456466       ME 15 Sec A
*/

#include<iostream>
using namespace std;

//declare matrices and result matrix
void multi ( int matrixA[3][3] ,int matrixB[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]+= matrixA[i][k] * matrixB[k][j];
         }
        }
    }
}

int main ()
{
    //declare matrices with elements
    int matrix1[3][3]={{31,22,13},{34,55,76},{72,18,79}};
    int matrix2[3][3]={{92,18,83},{63,59,14},{19,22,13}};
    int res[3][3];
    multi( matrix1,matrix2,res ); //calling function
    cout<<"The mulplication result of these matrices is: "<<endl;  //print
    for (int i=0;i<3;i++){
        for (int j=0;j<3;j++){
            cout<<res[i][j]<<" ";
        }
         cout<<endl;
    }
    return 0;
    }
```

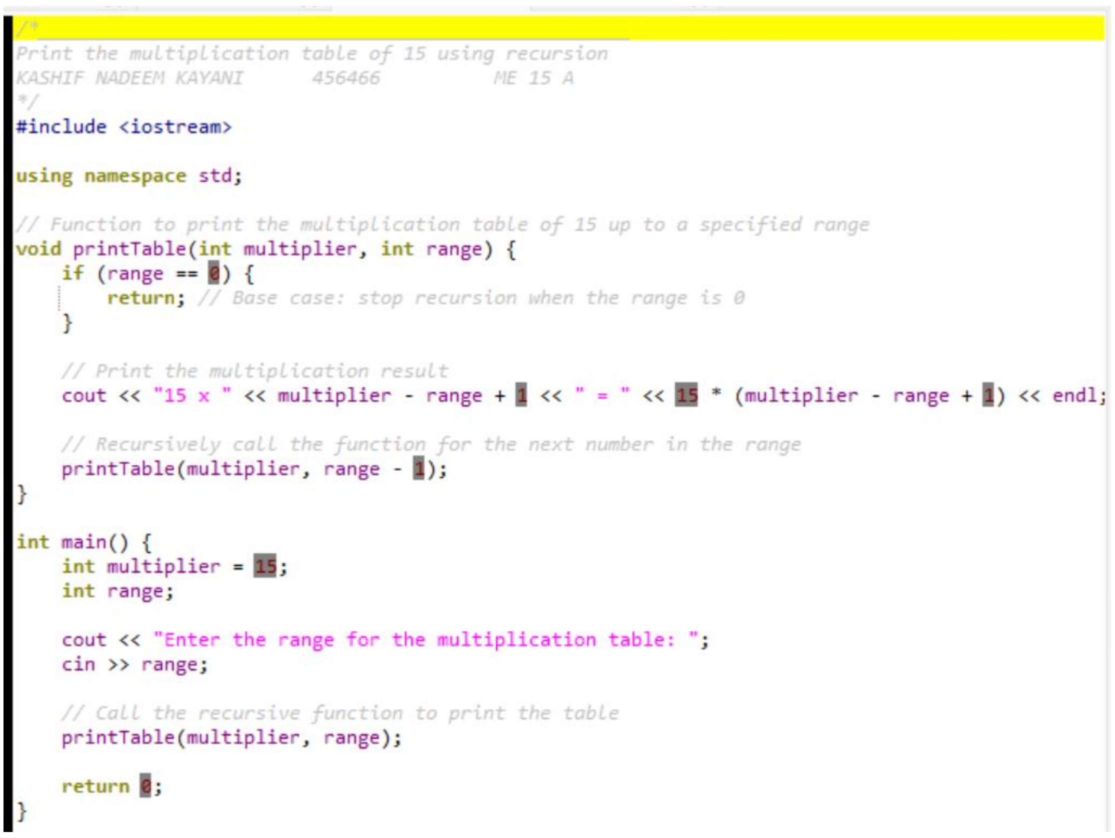
QUESTION NUMBER 05

```
/*______________________________________________________
Print the multiplication table of 15 using recursion
KASHIF NADEEM KAYANI          456466                ME 15 A
*/
#include <iostream>

using namespace std;

// Function to print the multiplication table of 15 up to a specified range
void printTable(int multiplier, int range) {
    if (range == 0) {
        return; // Base case: stop recursion when the range is 0
    }

    // Print the multiplication result
    cout << "15 x " << multiplier - range + 1 << " = " << 15 * (multiplier - range + 1) << endl;

    // Recursively call the function for the next number in the range
    printTable(multiplier, range - 1);
}

int main() {
    int multiplier = 15;
    int range;

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

    // Call the recursive function to print the table
    printTable(multiplier, range);

    return 0;
}
```



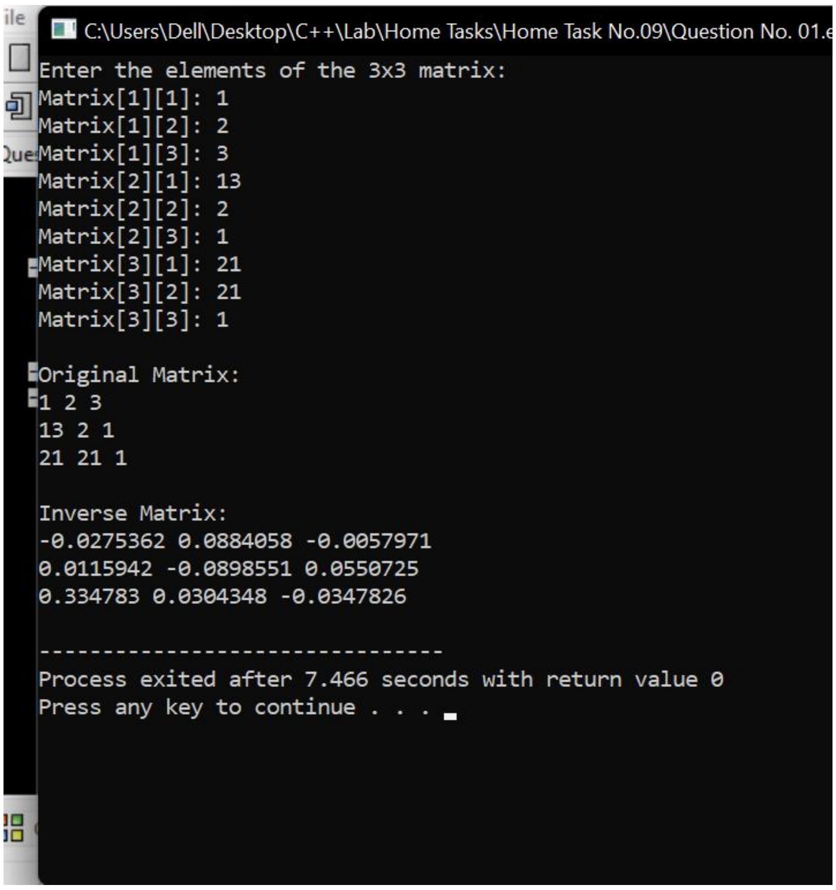
C:\Users\Dell\Desktop\C++\Lab\Lab tasks\Task No.09\Question No. 05.exe

```
Enter the range for the multiplication table: 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
15 x 11 = 165
15 x 12 = 180
15 x 13 = 195
15 x 14 = 210
15 x 15 = 225

--------------------------------
Process exited after 3.306 seconds with return value 0
Press any key to continue . . .
```


```cpp
/*
Print the multiplication table of 15 using recursion
KASHIF NADEEM KAYANI      456466          ME 15 A
*/
#include <iostream>

using namespace std;

// Function to print the multiplication table of 15 up to a specified range
void printTable(int multiplier, int range) {
    if (range == 0) {
        return; // Base case: stop recursion when the range is 0
    }

    // Print the multiplication result
    cout << "15 x " << multiplier - range + 1 << " = " << 15 * (multiplier - range + 1) << endl;

    // Recursively call the function for the next number in the range
    printTable(multiplier, range - 1);
}

int main() {
    int multiplier = 15;
    int range;

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

    // Call the recursive function to print the table
    printTable(multiplier, range);

    return 0;
}
```

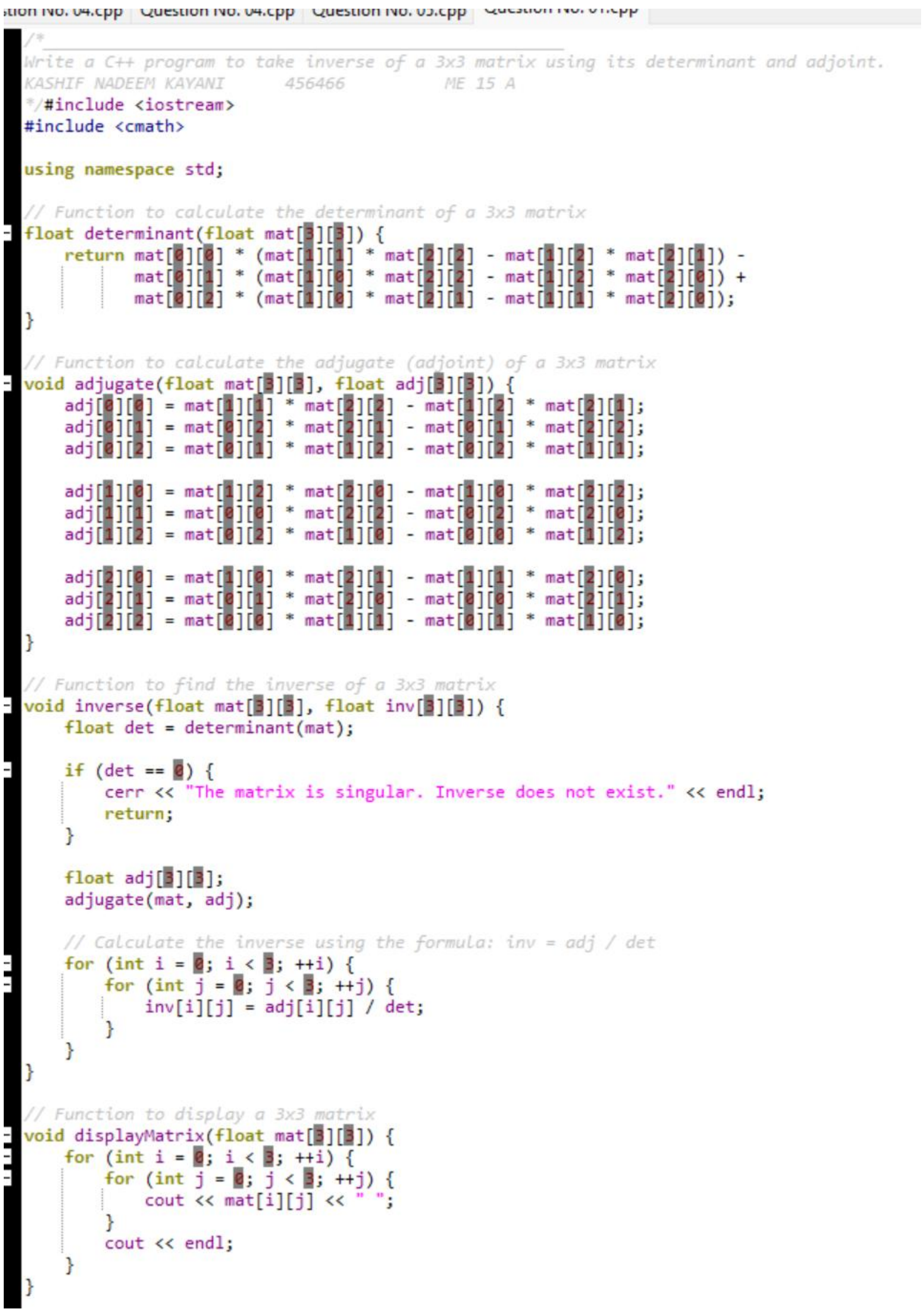
HOME TASK
QUESTION NUMBER 01

```
/*____________________________________________________
Write a C++ program to take inverse of a 3x3 matrix using its determinant and adjoint.
KASHIF NADEEM KAYANI          456466                 ME 15 A
*/#include <iostream>
#include <cmath>

using namespace std;

// Function to calculate the determinant of a 3x3 matrix
float determinant(float mat[3][3]) {
    return mat[0][0] * (mat[1][1] * mat[2][2] - mat[1][2] * mat[2][1]) -
           mat[0][1] * (mat[1][0] * mat[2][2] - mat[1][2] * mat[2][0]) +
           mat[0][2] * (mat[1][0] * mat[2][1] - mat[1][1] * mat[2][0]);
}

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

    adj[1][0] = mat[1][2] * mat[2][0] - mat[1][0] * mat[2][2];
    adj[1][1] = mat[0][0] * mat[2][2] - mat[0][2] * mat[2][0];
    adj[1][2] = mat[0][2] * mat[1][0] - mat[0][0] * mat[1][2];

    adj[2][0] = mat[1][0] * mat[2][1] - mat[1][1] * mat[2][0];
    adj[2][1] = mat[0][1] * mat[2][0] - mat[0][0] * mat[2][1];
    adj[2][2] = mat[0][0] * mat[1][1] - mat[0][1] * mat[1][0];
}

// Function to find the inverse of a 3x3 matrix
void inverse(float mat[3][3], float inv[3][3]) {
    float det = determinant(mat);

    if (det == 0) {
        cerr << "The matrix is singular. Inverse does not exist." << endl;
        return;
    }

    float adj[3][3];
    adjugate(mat, adj);

    // Calculate the inverse using the formula: inv = adj / det
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            inv[i][j] = adj[i][j] / det;
        }
    }
}

// Function to display a 3x3 matrix
void displayMatrix(float mat[3][3]) {
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << mat[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) {
            cout << "Matrix[" << i + 1 << "][" << j + 1 << "]: ";
            cin >> matrix[i][j];
        }
    }

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

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

    cout << "\nInverse Matrix:" << endl;
    displayMatrix(inverseMatrix);

    return 0;
}
```

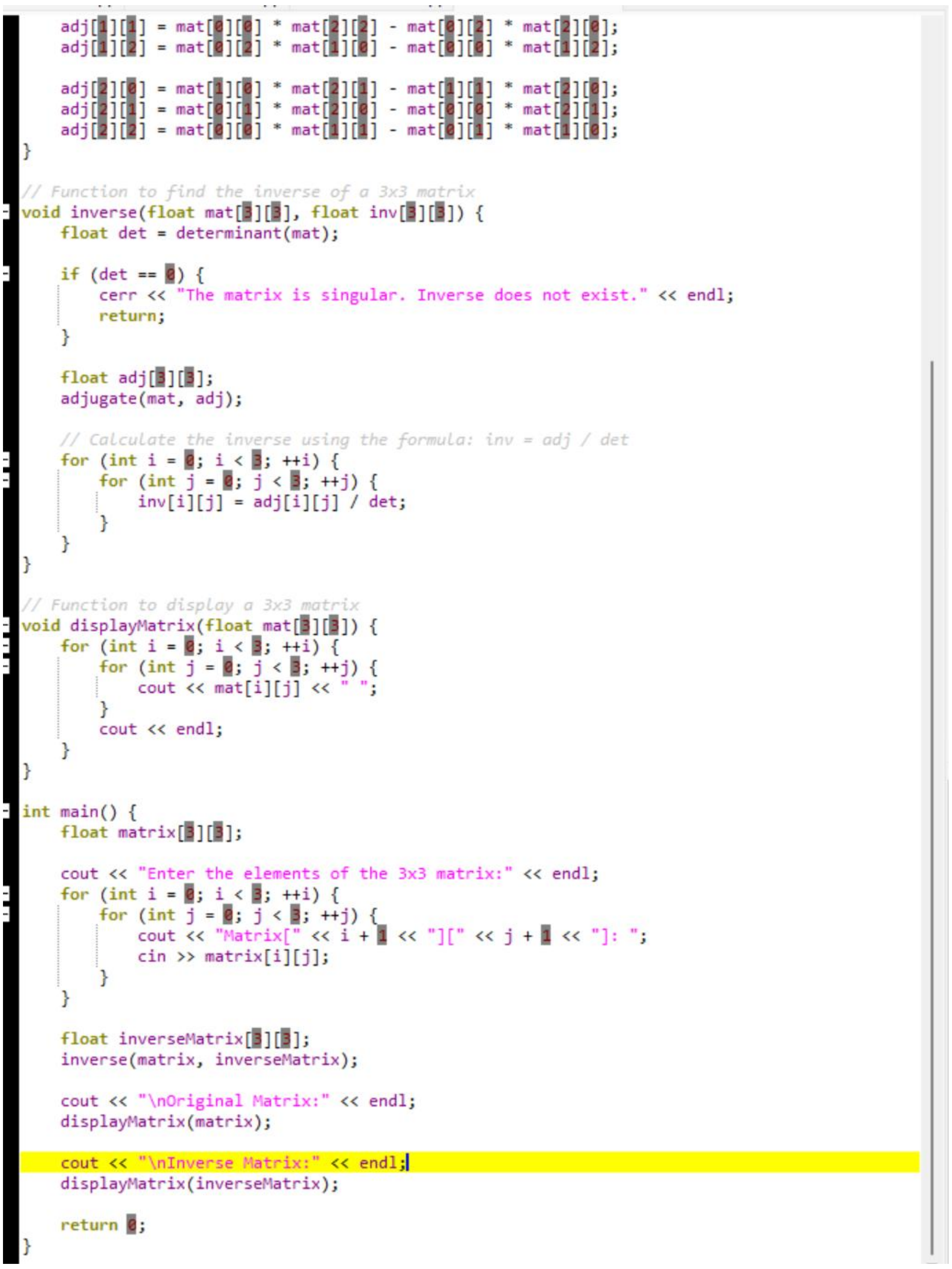


C:\Users\Dell\Desktop\C++\Lab\Home Tasks\Home Task No.09\Question No. 01.

```
Enter the elements of the 3x3 matrix:
Matrix[1][1]: 1
Matrix[1][2]: 2
Matrix[1][3]: 3
Matrix[2][1]: 13
Matrix[2][2]: 2
Matrix[2][3]: 1
Matrix[3][1]: 21
Matrix[3][2]: 21
Matrix[3][3]: 1

Original Matrix:
1 2 3
13 2 1
21 21 1

Inverse Matrix:
-0.0275362 0.0884058 -0.0057971
0.0115942 -0.0898551 0.0550725
0.334783 0.0304348 -0.0347826

--------------------------------
Process exited after 7.466 seconds with return value 0
Press any key to continue . . .
```



```cpp
/*_______________________________________________________________
Write a C++ program to take inverse of a 3x3 matrix using its determinant and adjoint.
KASHIF NADEEM KAYANI        456466          ME 15 A
*/#include <iostream>
#include <cmath>

using namespace std;

// Function to calculate the determinant of a 3x3 matrix
float determinant(float mat[3][3]) {
    return mat[0][0] * (mat[1][1] * mat[2][2] - mat[1][2] * mat[2][1]) -
           mat[0][1] * (mat[1][0] * mat[2][2] - mat[1][2] * mat[2][0]) +
           mat[0][2] * (mat[1][0] * mat[2][1] - mat[1][1] * mat[2][0]);
}

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

    adj[1][0] = mat[1][2] * mat[2][0] - mat[1][0] * mat[2][2];
    adj[1][1] = mat[0][0] * mat[2][2] - mat[0][2] * mat[2][0];
    adj[1][2] = mat[0][2] * mat[1][0] - mat[0][0] * mat[1][2];

    adj[2][0] = mat[1][0] * mat[2][1] - mat[1][1] * mat[2][0];
    adj[2][1] = mat[0][1] * mat[2][0] - mat[0][0] * mat[2][1];
    adj[2][2] = mat[0][0] * mat[1][1] - mat[0][1] * mat[1][0];
}

// Function to find the inverse of a 3x3 matrix
void inverse(float mat[3][3], float inv[3][3]) {
    float det = determinant(mat);

    if (det == 0) {
        cerr << "The matrix is singular. Inverse does not exist." << endl;
        return;
    }

    float adj[3][3];
    adjugate(mat, adj);

    // Calculate the inverse using the formula: inv = adj / det
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            inv[i][j] = adj[i][j] / det;
        }
    }
}

// Function to display a 3x3 matrix
void displayMatrix(float mat[3][3]) {
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << mat[i][j] << " ";
        }
        cout << endl;
    }
}
```



```cpp
    adj[1][1] = mat[0][0] * mat[2][2] - mat[0][2] * mat[2][0];
    adj[1][2] = mat[0][2] * mat[1][0] - mat[0][0] * mat[1][2];

    adj[2][0] = mat[1][0] * mat[2][1] - mat[1][1] * mat[2][0];
    adj[2][1] = mat[0][1] * mat[2][0] - mat[0][0] * mat[2][1];
    adj[2][2] = mat[0][0] * mat[1][1] - mat[0][1] * mat[1][0];
}

// Function to find the inverse of a 3x3 matrix
void inverse(float mat[3][3], float inv[3][3]) {
    float det = determinant(mat);

    if (det == 0) {
        cerr << "The matrix is singular. Inverse does not exist." << endl;
        return;
    }

    float adj[3][3];
    adjugate(mat, adj);

    // Calculate the inverse using the formula: inv = adj / det
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            inv[i][j] = adj[i][j] / det;
        }
    }
}

// Function to display a 3x3 matrix
void displayMatrix(float mat[3][3]) {
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << mat[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) {
            cout << "Matrix[" << i + 1 << "][" << j + 1 << "]: ";
            cin >> matrix[i][j];
        }
    }

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

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

    cout << "\nInverse Matrix:" << endl;
    displayMatrix(inverseMatrix);

    return 0;
}
```