NUST-SMME-

CS-114 Fundamentals of Programming LAB MANUAL#09

Lab+Hometasks

Date of submission: - 12-12-2023

Name: - Mohammad Abdullah Tahseen

Qalam ID: - 462573

BE-ME15 Section: - A

Course Instructor: Dr. Jawad Khan

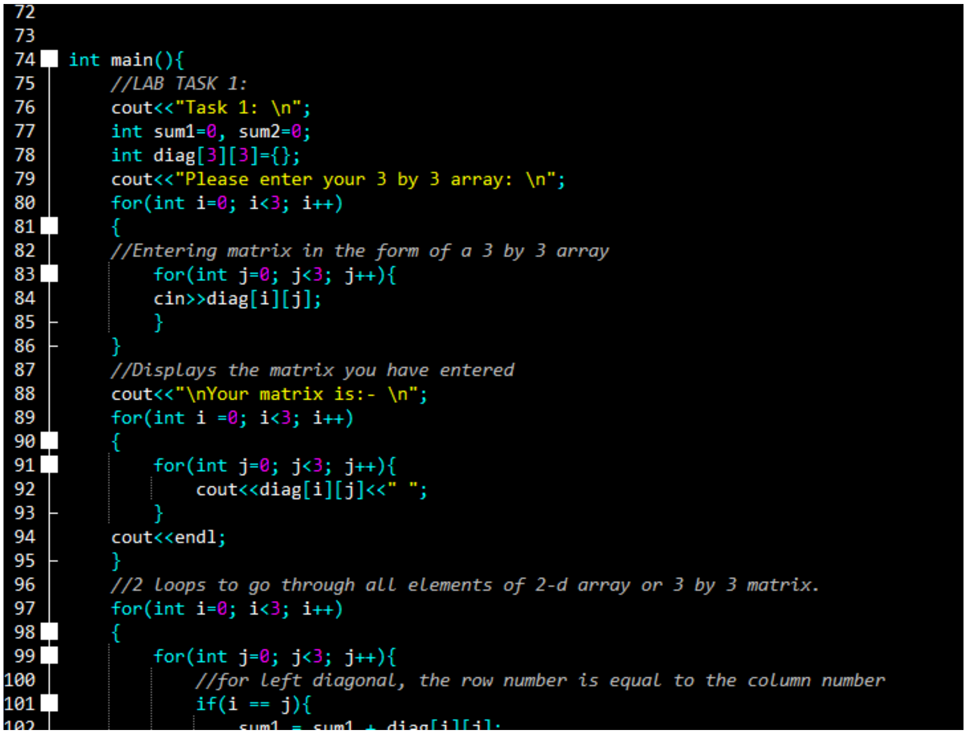
Lab Instructor: Muhammad Affan

LAB TASKS:

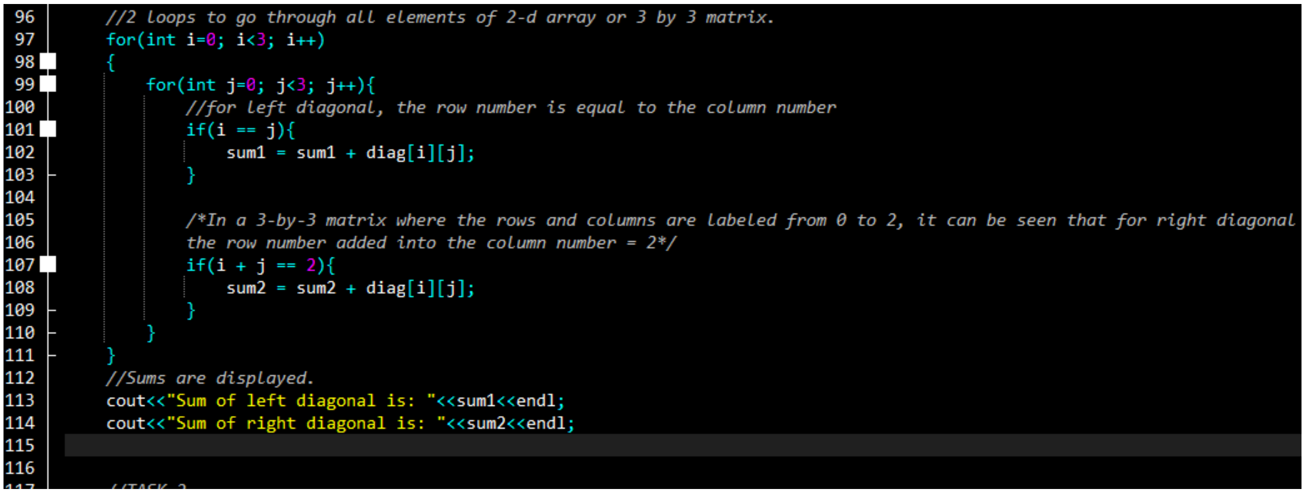
Task 1: -

Q1. Make 2D Array in C++ and print left diagonal and right diagonal sum of a 3 by 3 matrix

Code: -

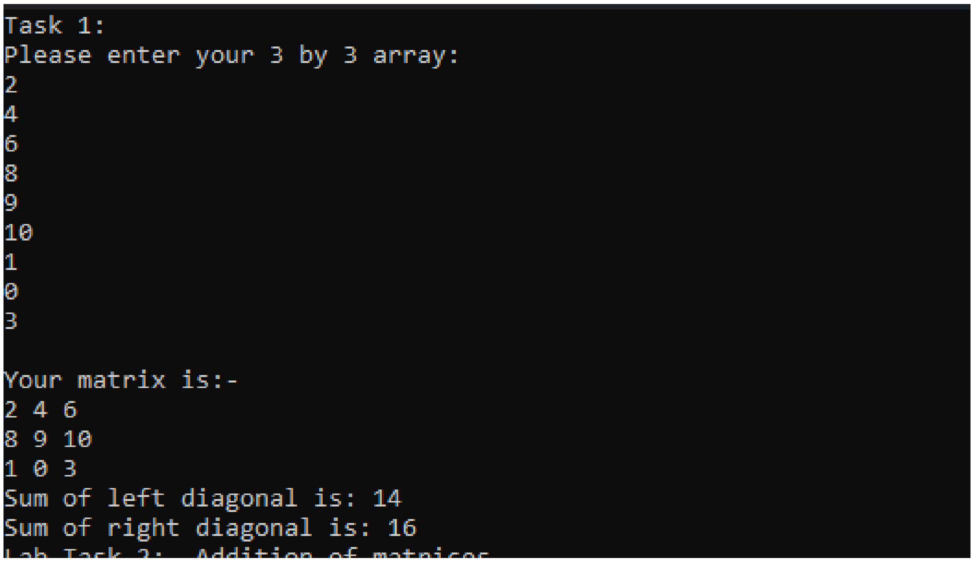


```cpp


int main(){
    //LAB TASK 1:
    cout<<"Task 1: \n";
    int sum1=0, sum2=0;
    int diag[3][3]={};
    cout<<"Please enter your 3 by 3 array: \n";
    for(int i=0; i<3; i++)
    {
    //Entering matrix in the form of a 3 by 3 array
        for(int j=0; j<3; j++){
        cin>>diag[i][j];
        }
    }
    //Displays the matrix you have entered
    cout<<"\nYour matrix is:- \n";
    for(int i =0; i<3; i++)
    {
        for(int j=0; j<3; j++){
            cout<<diag[i][j]<<" ";
        }
    cout<<endl;
    }
    //2 loops to go through all elements of 2-d array or 3 by 3 matrix.
    for(int i=0; i<3; i++)
    {
        for(int j=0; j<3; j++){
            //for left diagonal, the row number is equal to the column number
            if(i == j){
                sum1 = sum1 + diag[i][j];
```



```cpp
    //2 loops to go through all elements of 2-d array or 3 by 3 matrix.
    for(int i=0; i<3; i++)
    {
        for(int j=0; j<3; j++){
            //for left diagonal, the row number is equal to the column number
            if(i == j){
                sum1 = sum1 + diag[i][j];
            }

            /*In a 3-by-3 matrix where the rows and columns are labeled from 0 to 2, it can be seen that for right diagonal
            the row number added into the column number = 2*/
            if(i + j == 2){
                sum2 = sum2 + diag[i][j];
            }
        }
    }
    //Sums are displayed.
    cout<<"Sum of left diagonal is: "<<sum1<<endl;
    cout<<"Sum of right diagonal is: "<<sum2<<endl;


    //TASK 2
```

Output: -

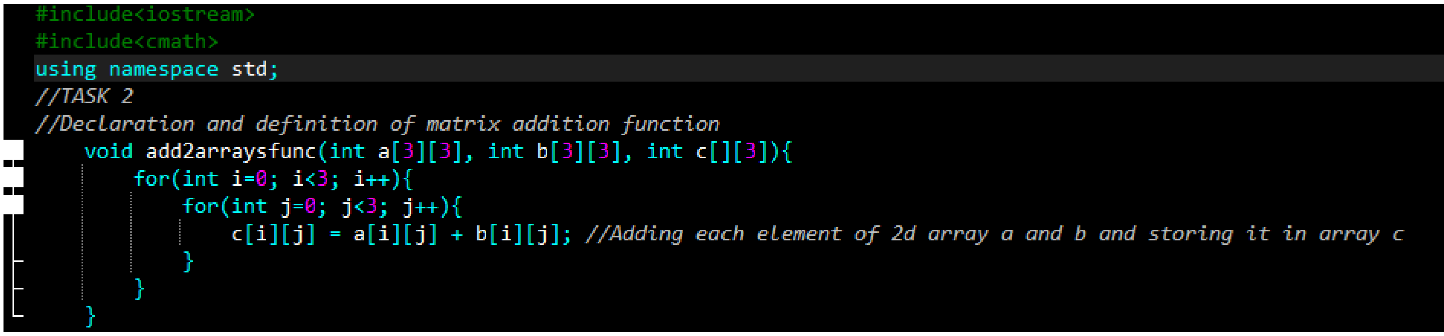


```
Task 1:
Please enter your 3 by 3 array:
2
4
6
8
9
10
1
0
3

Your matrix is:-
2 4 6
8 9 10
1 0 3
Sum of left diagonal is: 14
Sum of right diagonal is: 16
Lab Task 2:  Addition of matrices
```

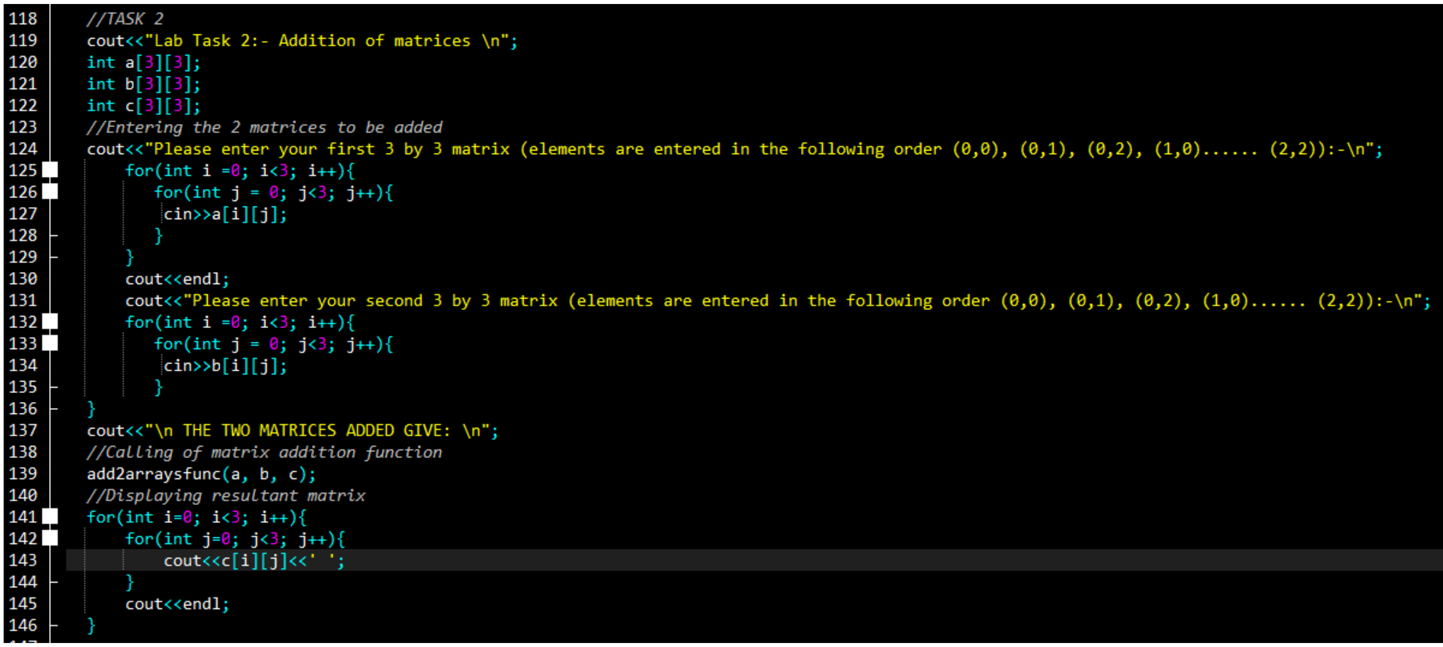
Task 2: -

Q2. Write a function to add two 2D arrays of 3 x 3.

Code: -

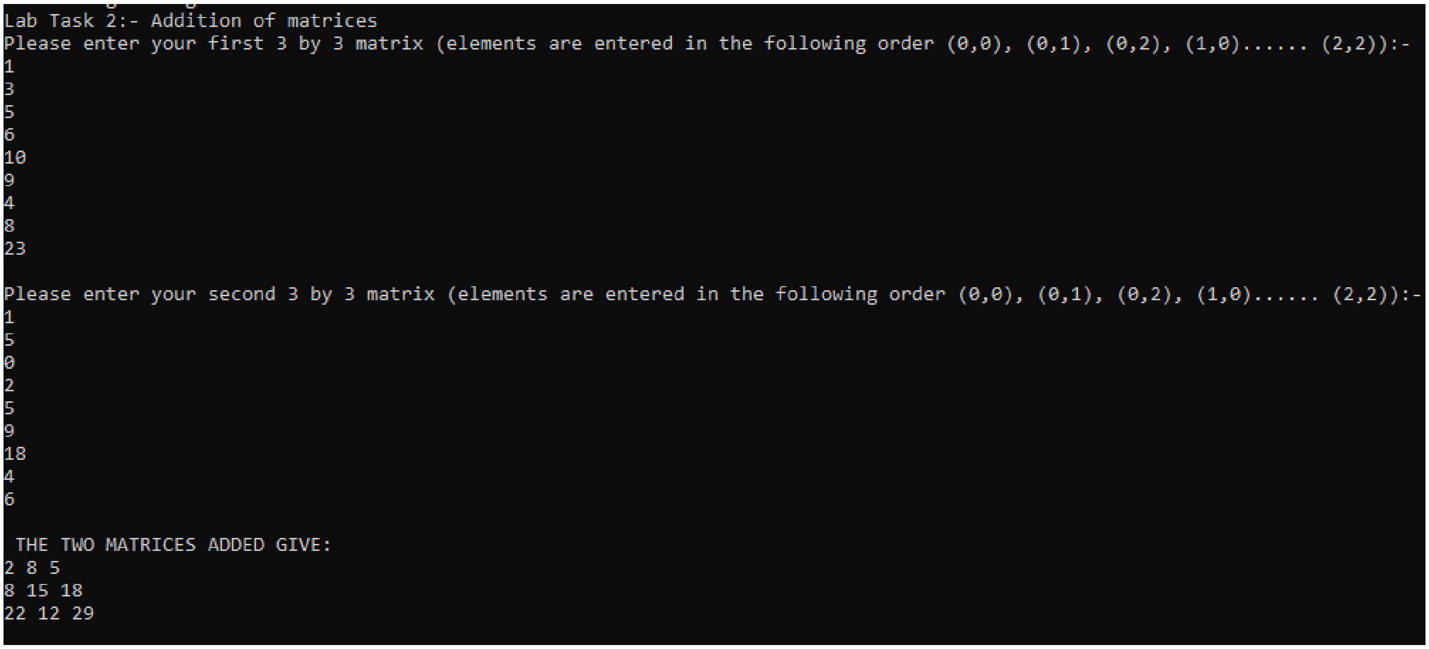


```
#include<iostream>
#include<cmath>
using namespace std;
//TASK 2
//Declaration and definition of matrix addition function
    void add2arraysfunc(int a[3][3], int b[3][3], int c[][3]){
        for(int i=0; i<3; i++){
            for(int j=0; j<3; j++){
                c[i][j] = a[i][j] + b[i][j]; //Adding each element of 2d array a and b and storing it in array c
            }
        }
    }
```



```
//TASK 2
cout<<"Lab Task 2:- Addition of matrices \n";
int a[3][3];
int b[3][3];
int c[3][3];
//Entering the 2 matrices to be added
cout<<"Please enter your first 3 by 3 matrix (elements are entered in the following order (0,0), (0,1), (0,2), (1,0)...... (2,2)):-\n";
    for(int i =0; i<3; i++){
       for(int j = 0; j<3; j++){
        cin>>a[i][j];
       }
    }
    cout<<endl;
    cout<<"Please enter your second 3 by 3 matrix (elements are entered in the following order (0,0), (0,1), (0,2), (1,0)...... (2,2)):-\n";
    for(int i =0; i<3; i++){
       for(int j = 0; j<3; j++){
        cin>>b[i][j];
       }
}
cout<<"\n THE TWO MATRICES ADDED GIVE: \n";
//Calling of matrix addition function
add2arraysfunc(a, b, c);
//Displaying resultant matrix
for(int i=0; i<3; i++){
    for(int j=0; j<3; j++){
        cout<<c[i][j]<<' ';
    }
    cout<<endl;
}
```

Output: -

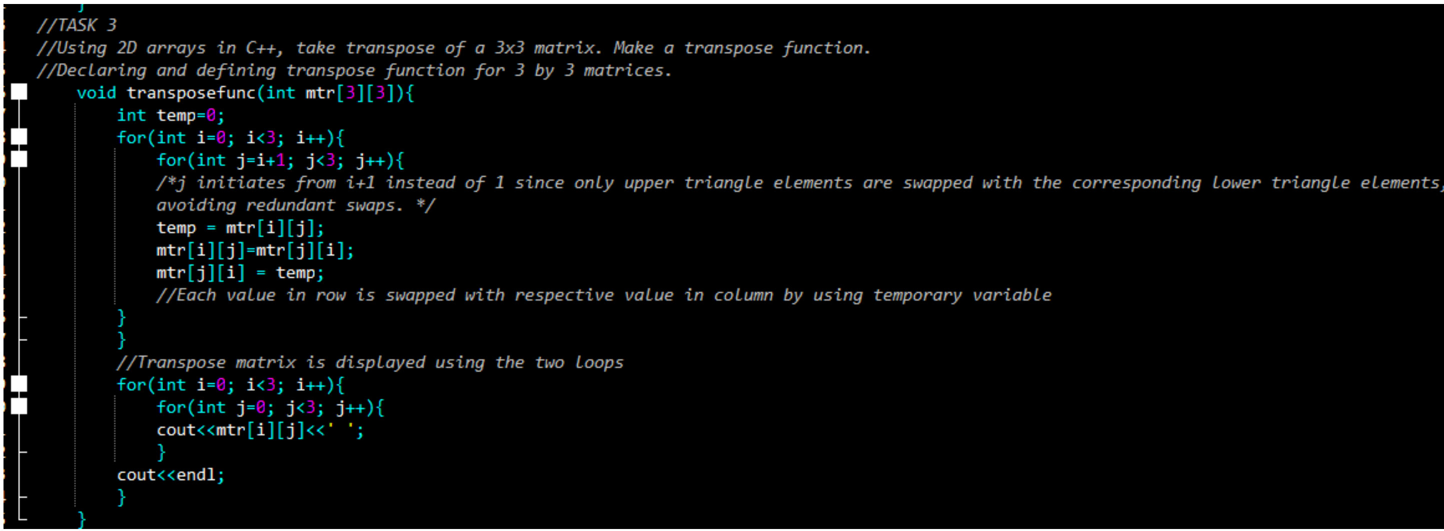


```
Lab Task 2:- Addition of matrices
Please enter your first 3 by 3 matrix (elements are entered in the following order (0,0), (0,1), (0,2), (1,0)...... (2,2)):-
1
3
5
6
10
9
4
8
23

Please enter your second 3 by 3 matrix (elements are entered in the following order (0,0), (0,1), (0,2), (1,0)...... (2,2)):-
1
5
0
2
5
9
18
4
6

 THE TWO MATRICES ADDED GIVE:
2 8 5
8 15 18
22 12 29
```

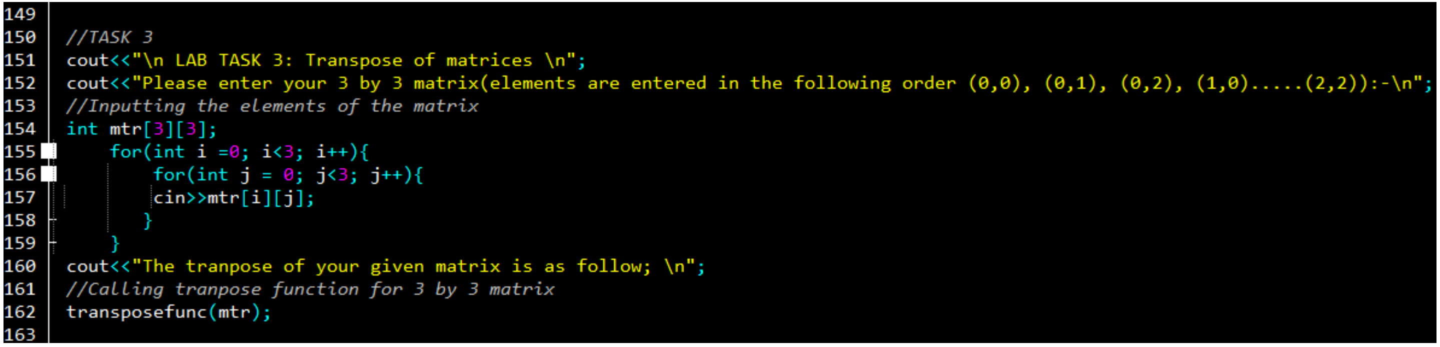
Task 3: -

Q3. Using 2D arrays in C++, take transpose of a 3 x 3 matrix. Make a transpose function.

Code: -

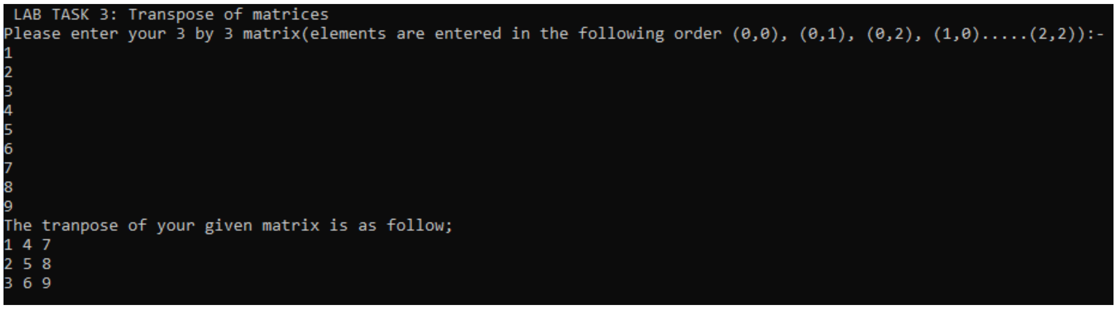


```
//TASK 3
//Using 2D arrays in C++, take transpose of a 3x3 matrix. Make a transpose function.
//Declaring and defining transpose function for 3 by 3 matrices.
    void transposefunc(int mtr[3][3]){
        int temp=0;
        for(int i=0; i<3; i++){
            for(int j=i+1; j<3; j++){
            /*j initiates from i+1 instead of 1 since only upper triangle elements are swapped with the corresponding lower triangle elements,
            avoiding redundant swaps. */
            temp = mtr[i][j];
            mtr[i][j]=mtr[j][i];
            mtr[j][i] = temp;
            //Each value in row is swapped with respective value in column by using temporary variable
        }
        }
        //Transpose matrix is displayed using the two loops
        for(int i=0; i<3; i++){
            for(int j=0; j<3; j++){
            cout<<mtr[i][j]<<' ';
            }
        cout<<endl;
        }
    }
```



```

//TASK 3
cout<<"\n LAB TASK 3: Transpose of matrices \n";
cout<<"Please enter your 3 by 3 matrix(elements are entered in the following order (0,0), (0,1), (0,2), (1,0).....(2,2)):-\n";
//Inputting the elements of the matrix
int mtr[3][3];
    for(int i =0; i<3; i++){
        for(int j = 0; j<3; j++){
        cin>>mtr[i][j];
        }
    }
cout<<"The tranpose of your given matrix is as follow; \n";
//Calling tranpose function for 3 by 3 matrix
transposefunc(mtr);

```

Output: -

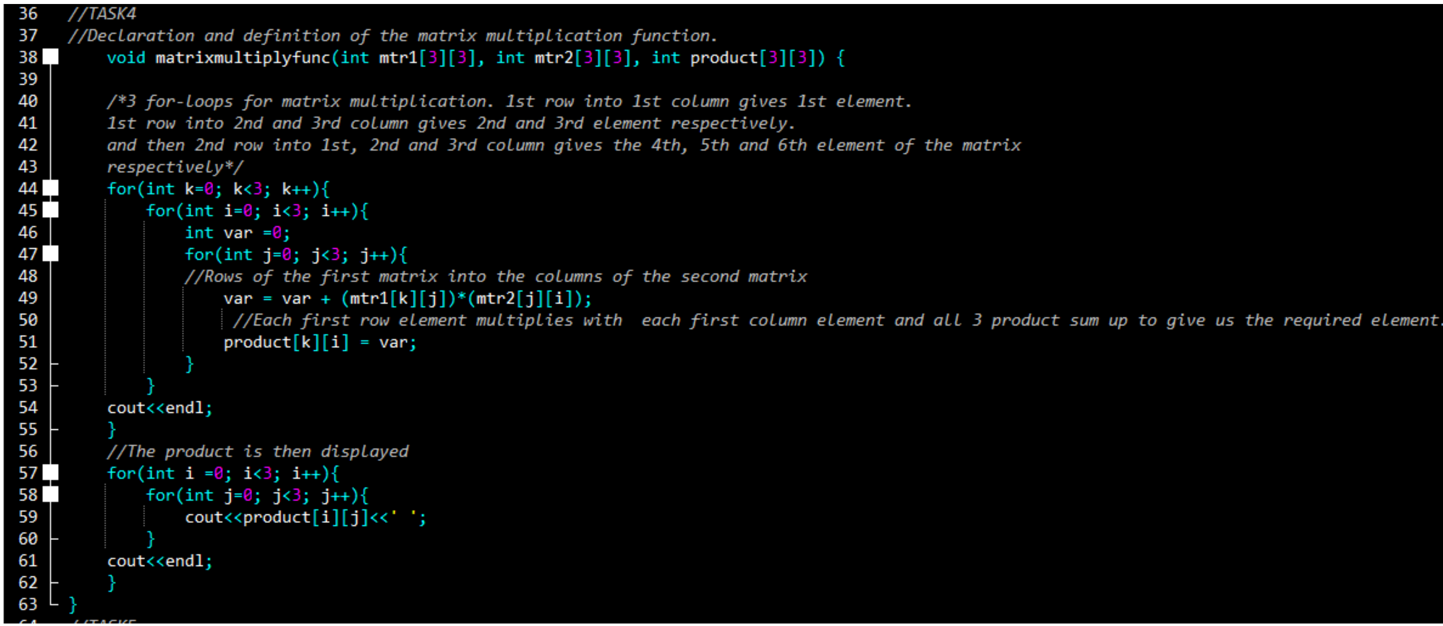


```
 LAB TASK 3: Transpose of matrices
Please enter your 3 by 3 matrix(elements are entered in the following order (0,0), (0,1), (0,2), (1,0).....(2,2)):-
1
2
3
4
5
6
7
8
9
The tranpose of your given matrix is as follow;
1 4 7
2 5 8
3 6 9
```

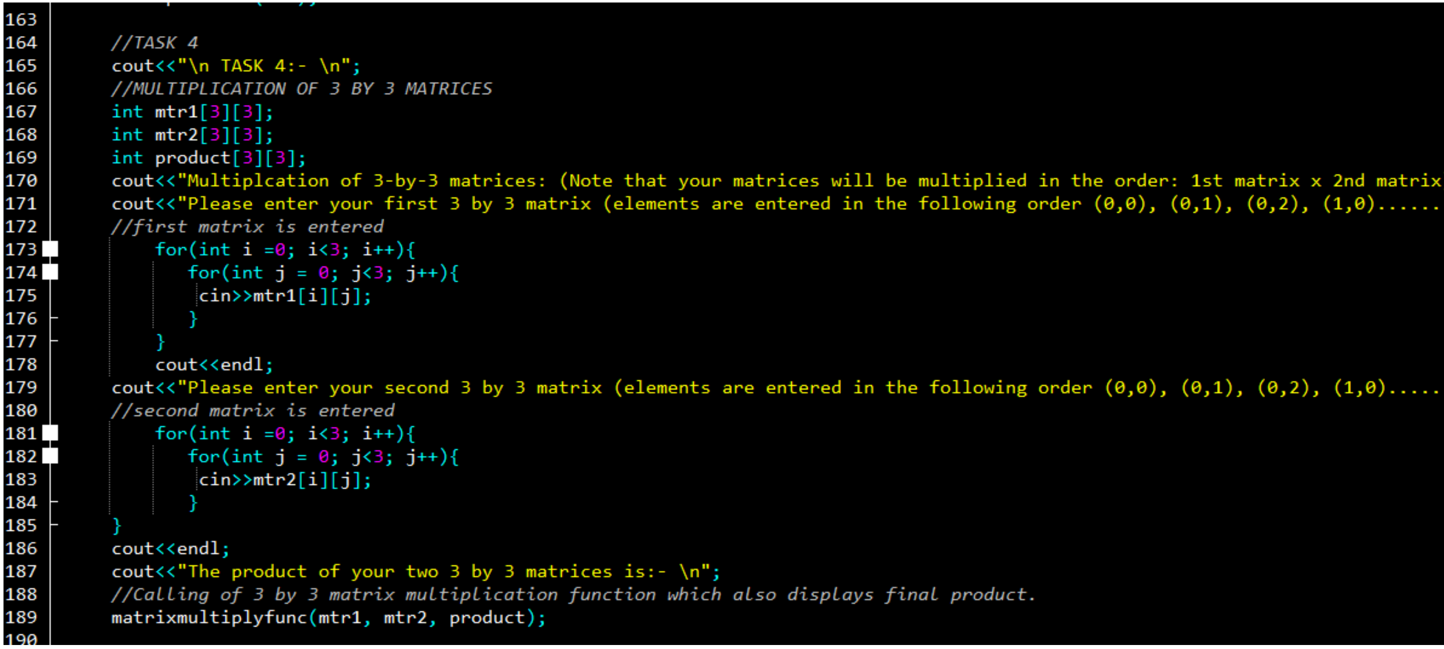
Task 4: -

Q4. Using 2D arrays in C++, implement 3 x 3 matrix multiplication. Make a function.

Code: -

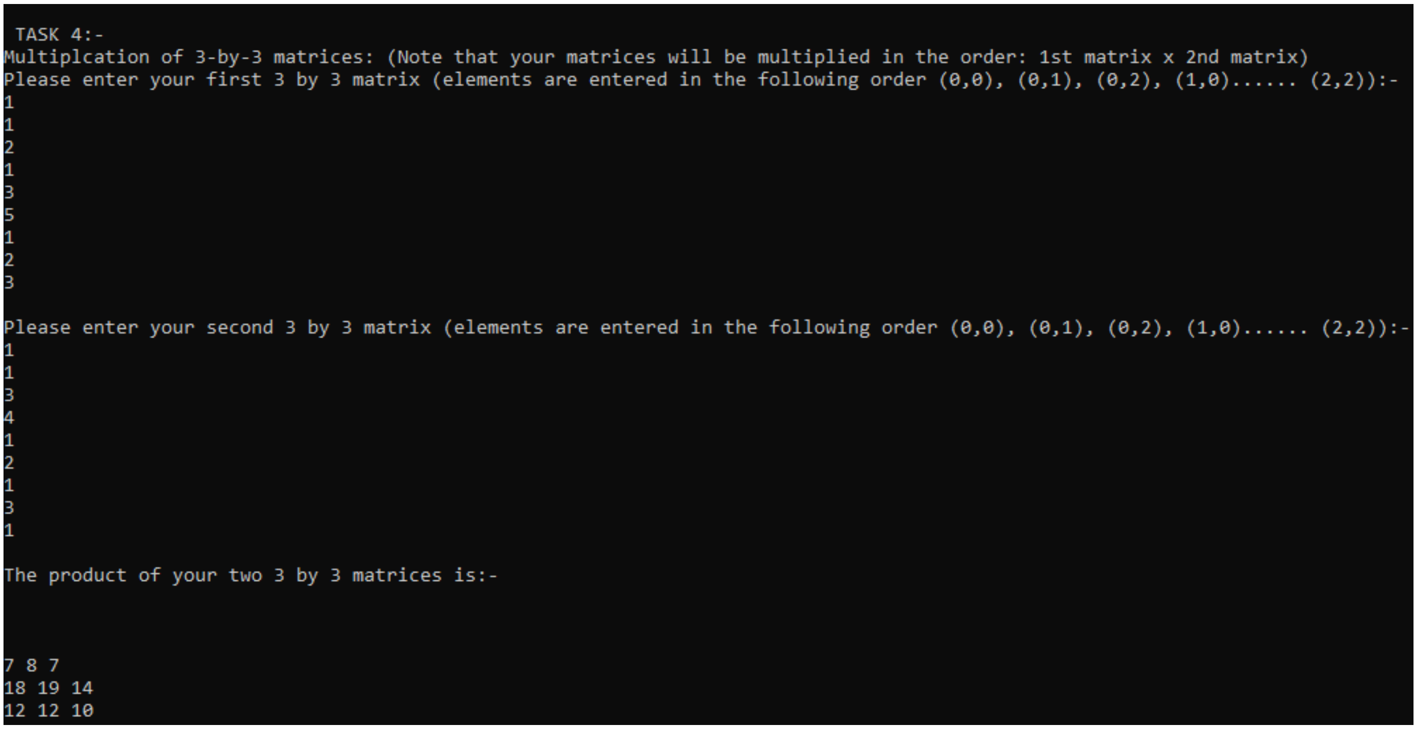


```
//TASK4
//Declaration and definition of the matrix multiplication function.
    void matrixmultiplyfunc(int mtr1[3][3], int mtr2[3][3], int product[3][3]) {

    /*3 for-loops for matrix multiplication. 1st row into 1st column gives 1st element.
    1st row into 2nd and 3rd column gives 2nd and 3rd element respectively.
    and then 2nd row into 1st, 2nd and 3rd column gives the 4th, 5th and 6th element of the matrix
    respectively*/
    for(int k=0; k<3; k++){
        for(int i=0; i<3; i++){
            int var =0;
            for(int j=0; j<3; j++){
            //Rows of the first matrix into the columns of the second matrix
                var = var + (mtr1[k][j])*(mtr2[j][i]);
                //Each first row element multiplies with  each first column element and all 3 product sum up to give us the required element.
                product[k][i] = var;
            }
        }
    cout<<endl;
    }
    //The product is then displayed
    for(int i =0; i<3; i++){
        for(int j=0; j<3; j++){
            cout<<product[i][j]<<' ';
        }
    cout<<endl;
    }
}
```



```

    //TASK 4
    cout<<"\n TASK 4:- \n";
    //MULTIPLICATION OF 3 BY 3 MATRICES
    int mtr1[3][3];
    int mtr2[3][3];
    int product[3][3];
    cout<<"Multiplcation of 3-by-3 matrices: (Note that your matrices will be multiplied in the order: 1st matrix x 2nd matrix
    cout<<"Please enter your first 3 by 3 matrix (elements are entered in the following order (0,0), (0,1), (0,2), (1,0)......
    //first matrix is entered
        for(int i =0; i<3; i++){
            for(int j = 0; j<3; j++){
            cin>>mtr1[i][j];
            }
        }
        cout<<endl;
    cout<<"Please enter your second 3 by 3 matrix (elements are entered in the following order (0,0), (0,1), (0,2), (1,0).....
    //second matrix is entered
        for(int i =0; i<3; i++){
            for(int j = 0; j<3; j++){
            cin>>mtr2[i][j];
            }
    }
    cout<<endl;
    cout<<"The product of your two 3 by 3 matrices is:- \n";
    //Calling of 3 by 3 matrix multiplication function which also displays final product.
    matrixmultiplyfunc(mtr1, mtr2, product);
```

Output: -

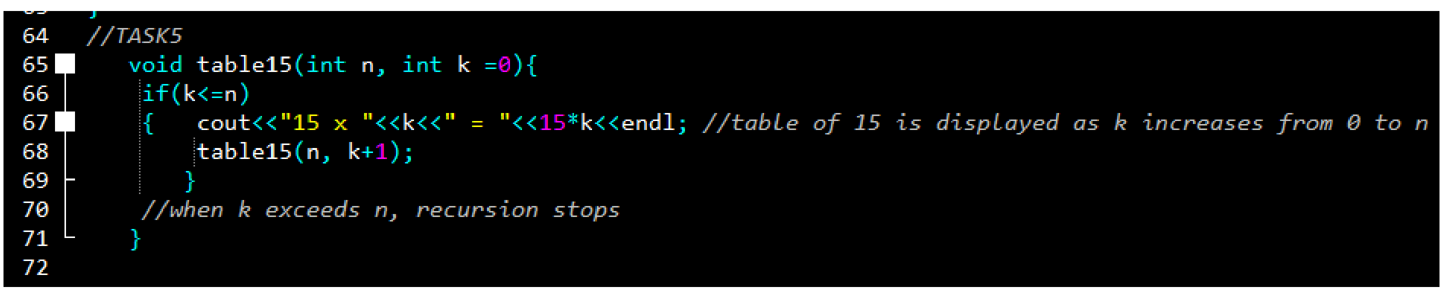


```
 TASK 4:-
Multiplcation of 3-by-3 matrices: (Note that your matrices will be multiplied in the order: 1st matrix x 2nd matrix)
Please enter your first 3 by 3 matrix (elements are entered in the following order (0,0), (0,1), (0,2), (1,0)...... (2,2)):-
1
1
2
1
3
5
1
2
3

Please enter your second 3 by 3 matrix (elements are entered in the following order (0,0), (0,1), (0,2), (1,0)...... (2,2)):-
1
1
3
4
1
2
1
3
1

The product of your two 3 by 3 matrices is:-



7 8 7
18 19 14
12 12 10
```

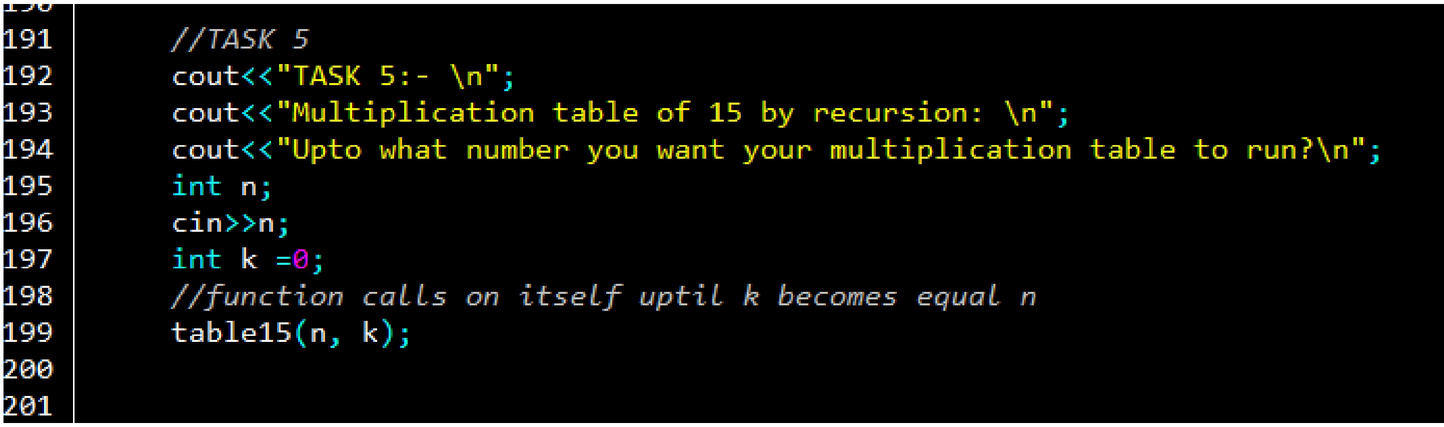
Task 5: -

Q5. Print the multiplication table of 15 using recursion.

Code: -

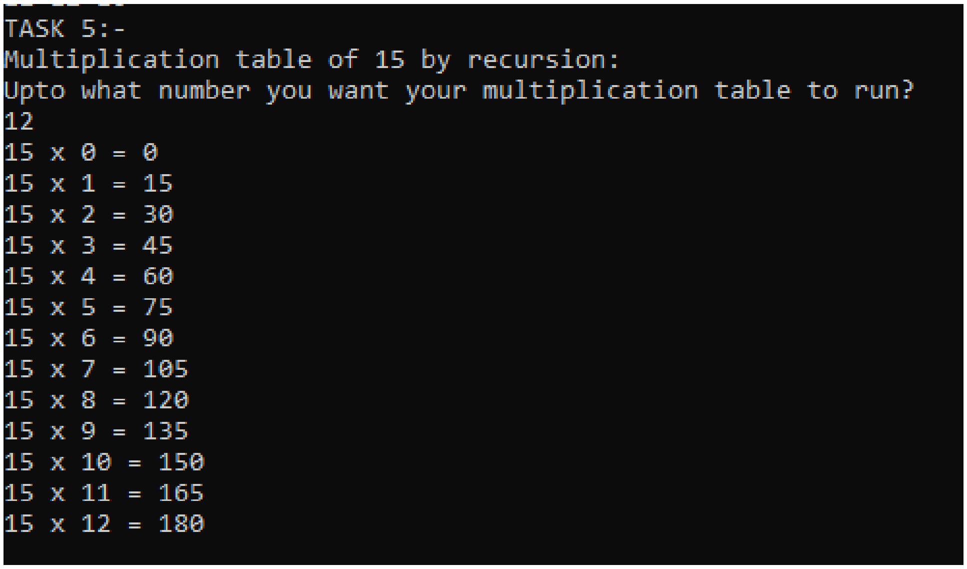


```
//TASK5
   void table15(int n, int k =0){
    if(k<=n)
    {    cout<<"15 x "<<k<<" = "<<15*k<<endl; //table of 15 is displayed as k increases from 0 to n
         table15(n, k+1);
        }
    //when k exceeds n, recursion stops
   }

```



```
    //TASK 5
    cout<<"TASK 5:- \n";
    cout<<"Multiplication table of 15 by recursion: \n";
    cout<<"Upto what number you want your multiplication table to run?\n";
    int n;
    cin>>n;
    int k =0;
    //function calls on itself uptil k becomes equal n
    table15(n, k);


```

Output: -



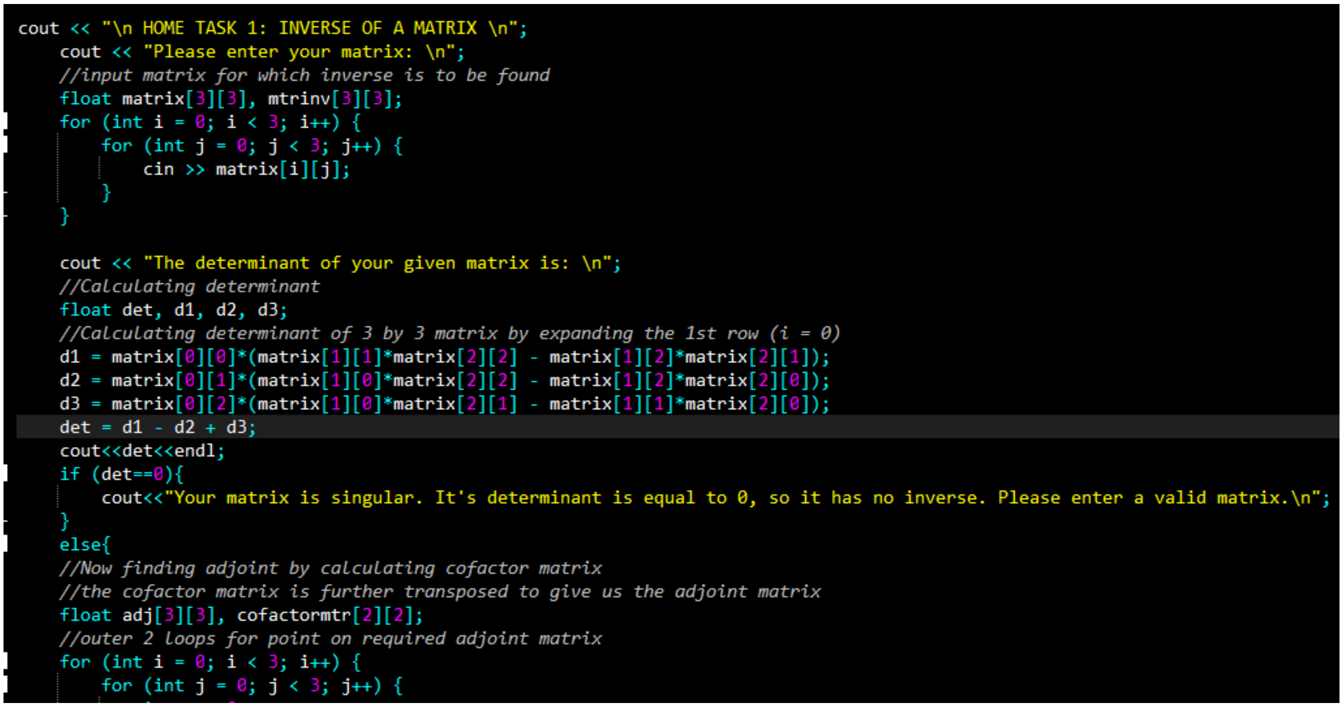
```
TASK 5:-
Multiplication table of 15 by recursion:
Upto what number you want your multiplication table to run?
12
15 x 0 = 0
15 x 1 = 15
15 x 2 = 30
15 x 3 = 45
15 x 4 = 60
15 x 5 = 75
15 x 6 = 90
15 x 7 = 105
15 x 8 = 120
15 x 9 = 135
15 x 10 = 150
15 x 11 = 165
15 x 12 = 180
```

HOME TASK:

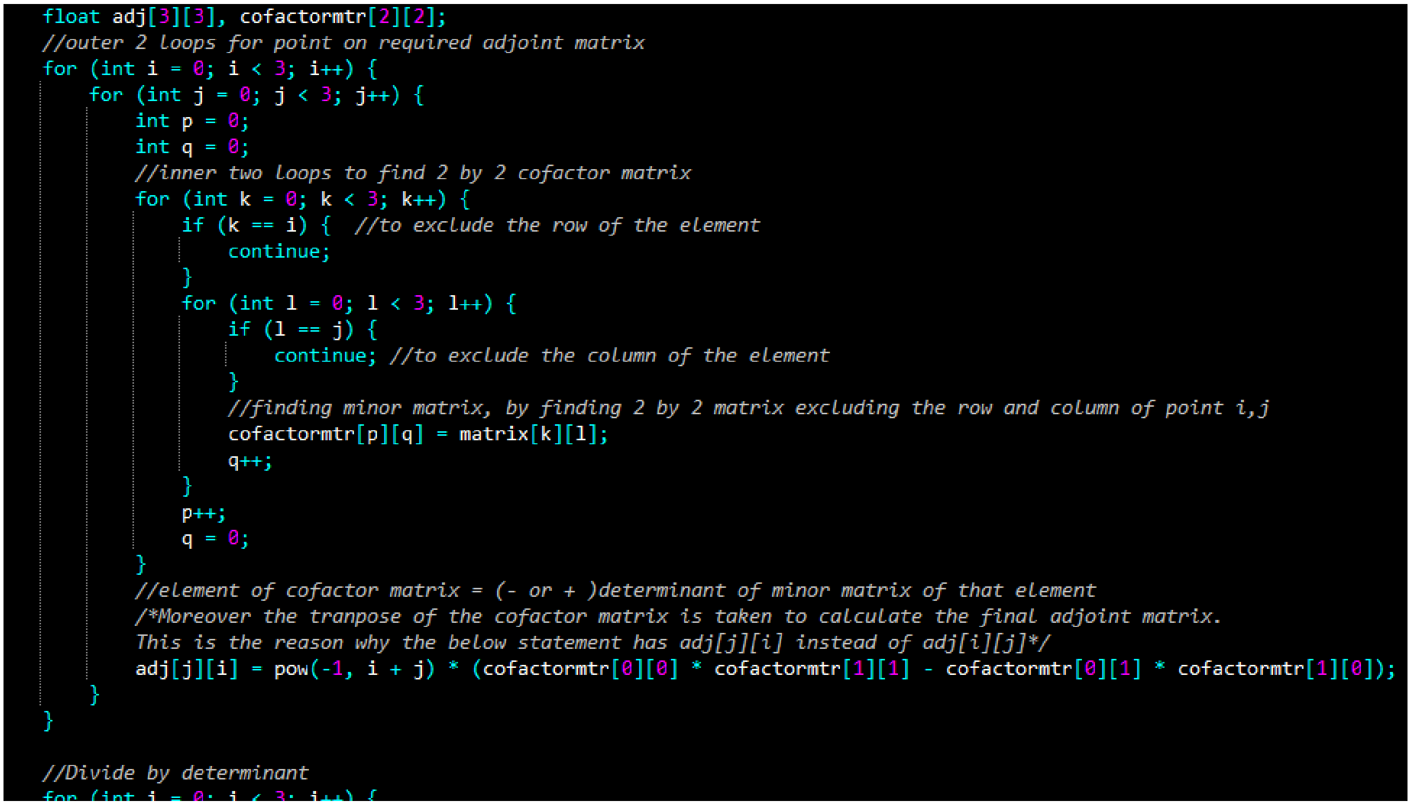
Task 1: -

Q1. Write a C++ program to take inverse of a 3 x 3 matrix using its determinant and adjoint.

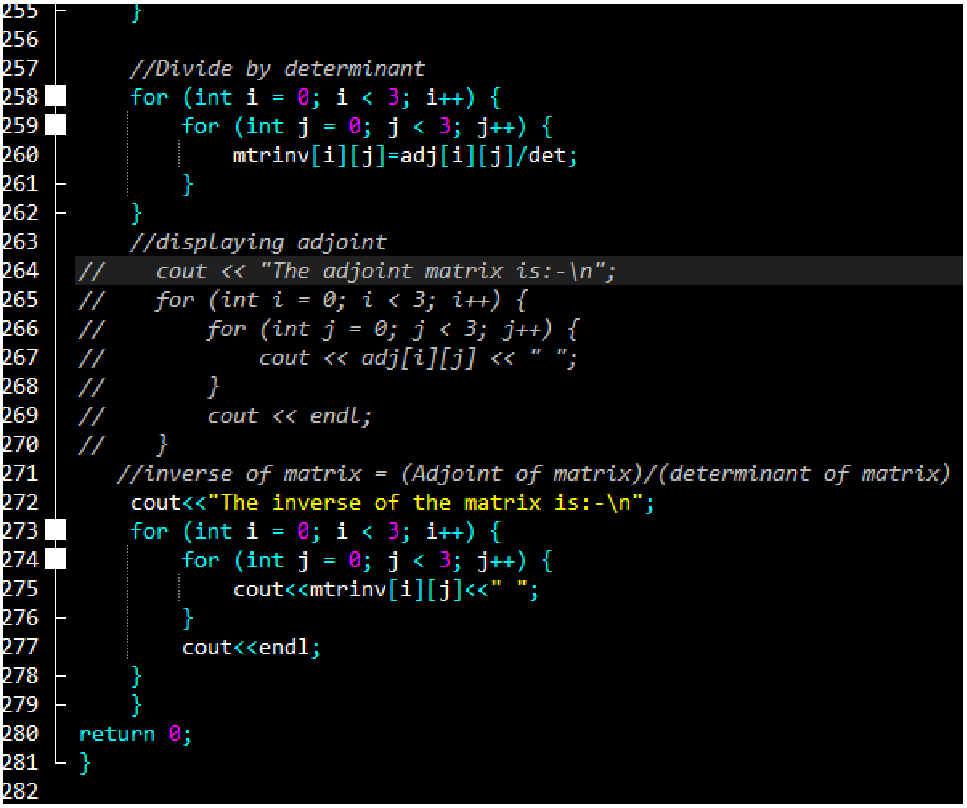
Code: -



```
cout << "\n HOME TASK 1: INVERSE OF A MATRIX \n";
    cout << "Please enter your matrix: \n";
    //input matrix for which inverse is to be found
    float matrix[3][3], mtrinv[3][3];
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cin >> matrix[i][j];
        }
    }

    cout << "The determinant of your given matrix is: \n";
    //Calculating determinant
    float det, d1, d2, d3;
    //Calculating determinant of 3 by 3 matrix by expanding the 1st row (i = 0)
    d1 = matrix[0][0]*(matrix[1][1]*matrix[2][2] - matrix[1][2]*matrix[2][1]);
    d2 = matrix[0][1]*(matrix[1][0]*matrix[2][2] - matrix[1][2]*matrix[2][0]);
    d3 = matrix[0][2]*(matrix[1][0]*matrix[2][1] - matrix[1][1]*matrix[2][0]);
    det = d1 - d2 + d3;
    cout<<det<<endl;
    if (det==0){
        cout<<"Your matrix is singular. It's determinant is equal to 0, so it has no inverse. Please enter a valid matrix.\n";
    }
    else{
    //Now finding adjoint by calculating cofactor matrix
    //the cofactor matrix is further transposed to give us the adjoint matrix
    float adj[3][3], cofactormtr[2][2];
    //outer 2 loops for point on required adjoint matrix
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
```

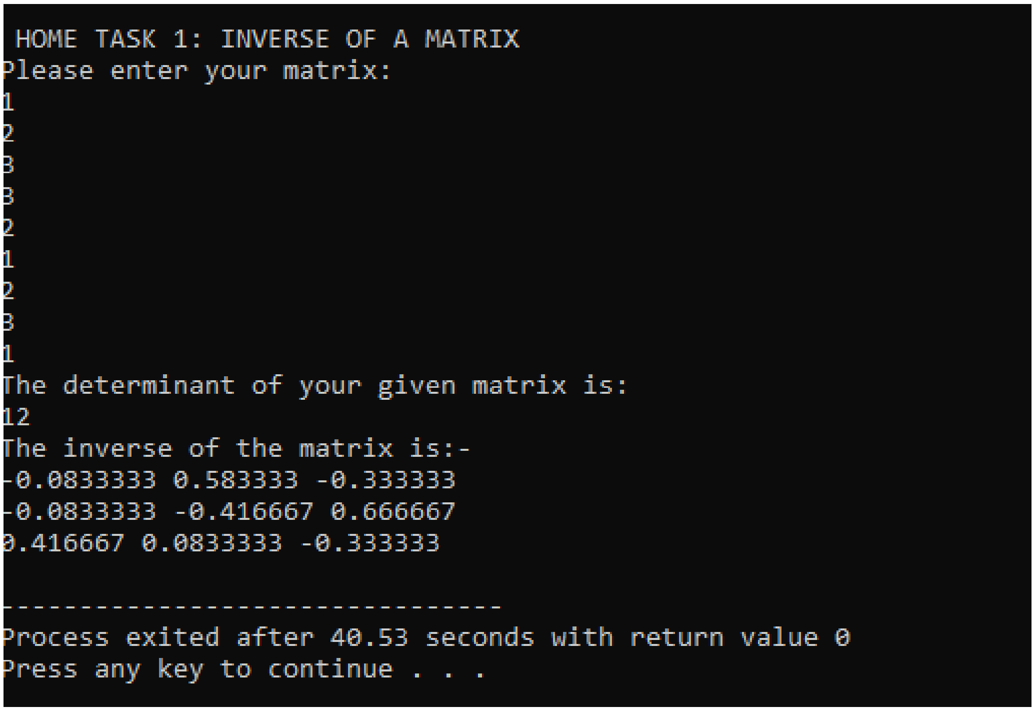


```
float adj[3][3], cofactormtr[2][2];
//outer 2 loops for point on required adjoint matrix
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
        int p = 0;
        int q = 0;
        //inner two loops to find 2 by 2 cofactor matrix
        for (int k = 0; k < 3; k++) {
            if (k == i) {  //to exclude the row of the element
                continue;
            }
            for (int l = 0; l < 3; l++) {
                if (l == j) {
                    continue; //to exclude the column of the element
                }
                //finding minor matrix, by finding 2 by 2 matrix excluding the row and column of point i,j
                cofactormtr[p][q] = matrix[k][l];
                q++;
            }
            p++;
            q = 0;
        }
        //element of cofactor matrix = (- or + )determinant of minor matrix of that element
        /*Moreover the tranpose of the cofactor matrix is taken to calculate the final adjoint matrix.
        This is the reason why the below statement has adj[j][i] instead of adj[i][j]*/
        adj[j][i] = pow(-1, i + j) * (cofactormtr[0][0] * cofactormtr[1][1] - cofactormtr[0][1] * cofactormtr[1][0]);
    }
}

//Divide by determinant
for (int i = 0; i < 3; i++) {
```

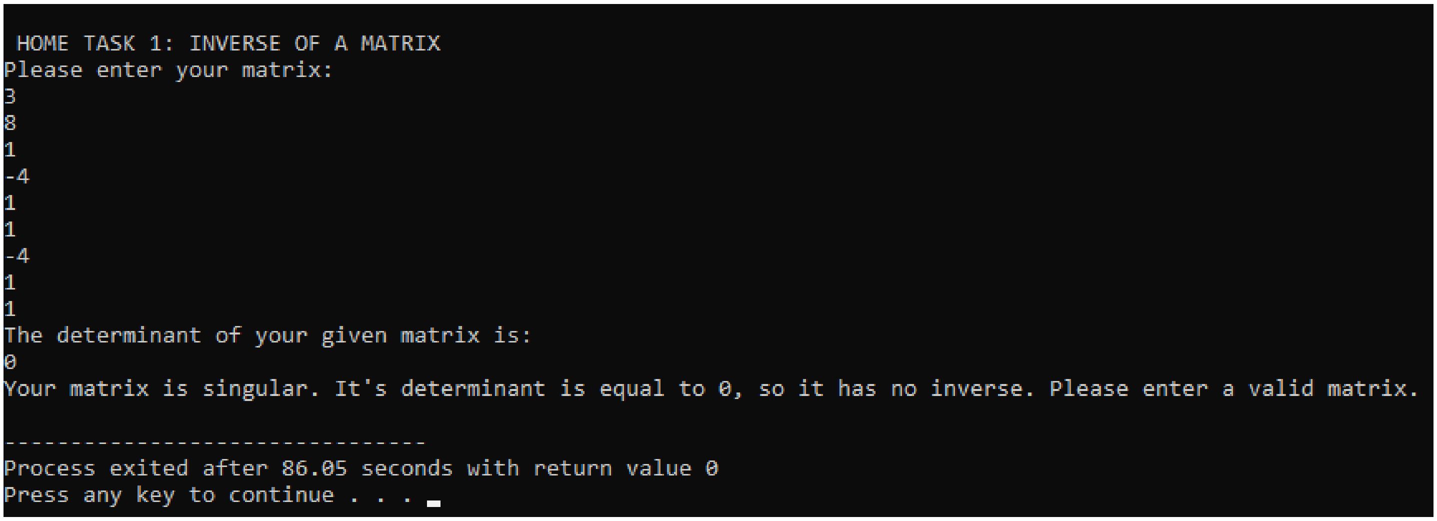


```
    }

    //Divide by determinant
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            mtrinv[i][j]=adj[i][j]/det;
        }
    }
    //displaying adjoint
//    cout << "The adjoint matrix is:-\n";
//    for (int i = 0; i < 3; i++) {
//        for (int j = 0; j < 3; j++) {
//            cout << adj[i][j] << " ";
//        }
//        cout << endl;
//    }
   //inverse of matrix = (Adjoint of matrix)/(determinant of matrix)
    cout<<"The inverse of the matrix is:-\n";
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout<<mtrinv[i][j]<<" ";
        }
        cout<<endl;
    }
    }
return 0;
}
```

Output: -



```
 HOME TASK 1: INVERSE OF A MATRIX
Please enter your matrix:
1
2
3
3
2
1
2
3
1
The determinant of your given matrix is:
12
The inverse of the matrix is:-
-0.0833333 0.583333 -0.333333
-0.0833333 -0.416667 0.666667
0.416667 0.0833333 -0.333333

--------------------------------
Process exited after 40.53 seconds with return value 0
Press any key to continue . . .
```



```
 HOME TASK 1: INVERSE OF A MATRIX
Please enter your matrix:
3
8
1
-4
1
1
-4
1
1
The determinant of your given matrix is:
0
Your matrix is singular. It's determinant is equal to 0, so it has no inverse. Please enter a valid matrix.

--------------------------------
Process exited after 86.05 seconds with return value 0
Press any key to continue . . .
```