NATIONAL UNIVERSITY OF SCIENCE AND TECHNOLOGY

CS-114- FUNDAMENTAL OF PROGRAMMING

LAB MANUAL 9 HOME TASKS

SUBMITTED BY:

TOSEEF HAIDER

SECTION:ME-15 (C)

CMS ID: 457249

SUBMITTED TO:

COURSE INSTRUCTOR: DR TALHA SHAHID

LAB INSTRUCTOR: MUHAMMAD AFFAN

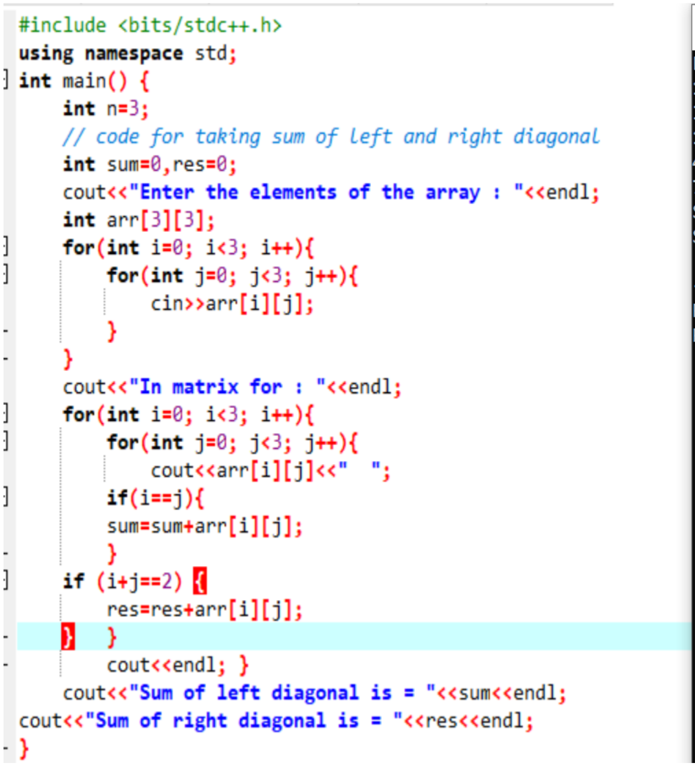
DATE OF SUBMISSION:

14 DECEMBER 2023

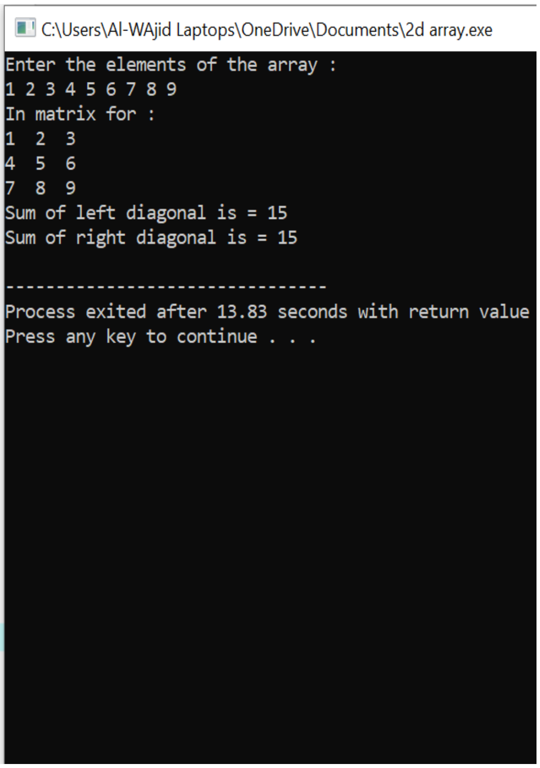
TASK 1:

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

CODE:



```
#include <bits/stdc++.h>
using namespace std;
int main() {
    int n=3;
    // code for taking sum of left and right diagonal
    int sum=0,res=0;
    cout<<"Enter the elements of the array : "<<endl;
    int arr[3][3];
    for(int i=0; i<3; i++){
        for(int j=0; j<3; j++){
            cin>>arr[i][j];
        }
    }
    cout<<"In matrix for : "<<endl;
    for(int i=0; i<3; i++){
        for(int j=0; j<3; j++){
            cout<<arr[i][j]<<"  ";
            if(i==j){
            sum=sum+arr[i][j];
            }
    if (i+j==2) {
            res=res+arr[i][j];
    }   }
            cout<<endl; }
    cout<<"Sum of left diagonal is = "<<sum<<endl;
cout<<"Sum of right diagonal is = "<<res<<endl;
}
```



C:\Users\Al-WAjid Laptops\OneDrive\Documents\2d array.exe

```
Enter the elements of the array :
1 2 3 4 5 6 7 8 9
In matrix for :
1  2  3
4  5  6
7  8  9
Sum of left diagonal is = 15
Sum of right diagonal is = 15

--------------------------------
Process exited after 13.83 seconds with return value
Press any key to continue . . .
```

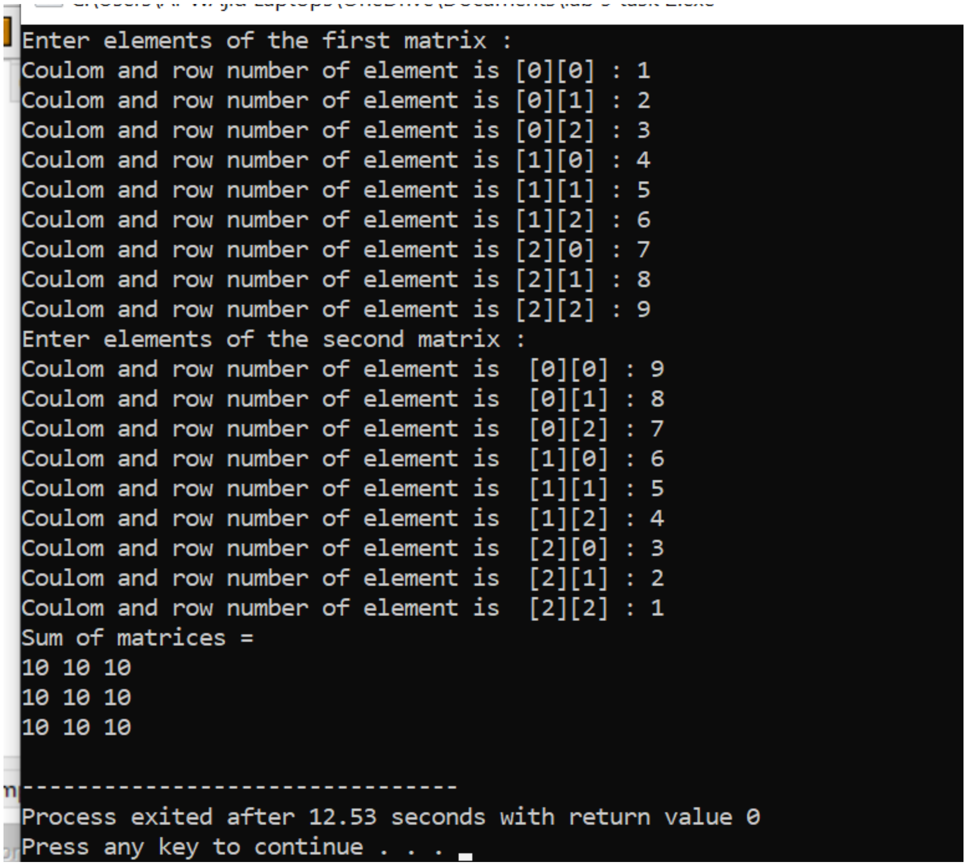
TASK 2: Write a function to add two 2D arrays of size 3x3.:

CODE:

```
#include <bits/stdc++.h>

using namespace std;

const int n = 3;

// Function to add two 3x3 matrices

void addmatrices(int mat1[n][n], int mat2[n][n], int result[n][n]) {

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

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

         result[i][j] = mat1[i][j] + mat2[i][j];

      }

   }  }
```

```
int main() {
    int mat1[n][n];
    int mat2[n][n];
    int result[n][n];
    cout << "Enter elements of the first matrix : "<<endl;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
              cout<<"Coulom and row number of element is ["<<i<<"]["<<j<<"] : ";
            cin >> mat1[i][j];
        }
    }
    cout << "Enter elements of the second matrix : "<<endl;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
              cout<<"Coulom and row number of element is  ["<<i<<"]["<<j<<"] : ";
            cin >> mat2[i][j];
        }
    }       // Call the function to add matrices
    addmatrices(mat1, mat2, result);
    // Display the result
    cout << "Sum of matrices = "<<endl;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cout << result[i][j] << " ";
        }
        cout << endl;  }
    return 0;
}
```

RESULT:



```
Enter elements of the first matrix :
Coulom and row number of element is [0][0] : 1
Coulom and row number of element is [0][1] : 2
Coulom and row number of element is [0][2] : 3
Coulom and row number of element is [1][0] : 4
Coulom and row number of element is [1][1] : 5
Coulom and row number of element is [1][2] : 6
Coulom and row number of element is [2][0] : 7
Coulom and row number of element is [2][1] : 8
Coulom and row number of element is [2][2] : 9
Enter elements of the second matrix :
Coulom and row number of element is  [0][0] : 9
Coulom and row number of element is  [0][1] : 8
Coulom and row number of element is  [0][2] : 7
Coulom and row number of element is  [1][0] : 6
Coulom and row number of element is  [1][1] : 5
Coulom and row number of element is  [1][2] : 4
Coulom and row number of element is  [2][0] : 3
Coulom and row number of element is  [2][1] : 2
Coulom and row number of element is  [2][2] : 1
Sum of matrices =
10 10 10
10 10 10
10 10 10

--------------------------------
Process exited after 12.53 seconds with return value 0
Press any key to continue . . .
```

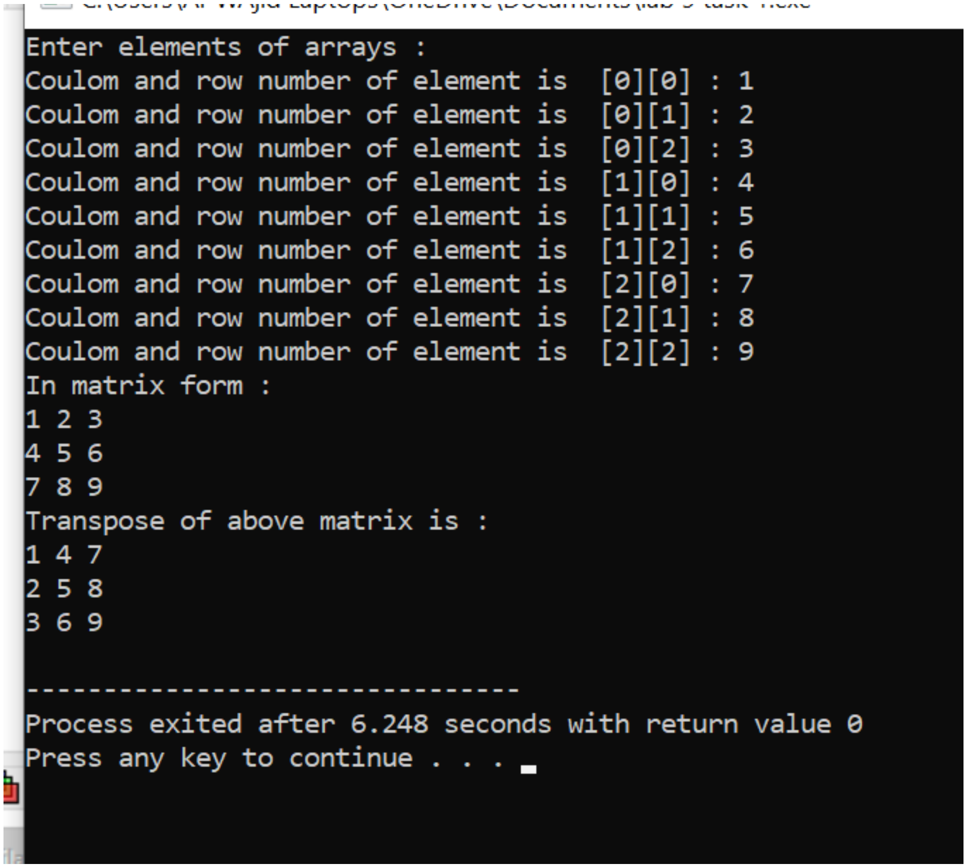
TASK 3:

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

CODE:

```
#include<bits/stdc++.h>

using namespace std;

const int n=3;

void transpose (int arr[n][n]) {

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

for( int j=i;j<n;j++) {

int temp = arr[i][j];

arr[i][j]=arr[j][i];

arr[j][i]=temp; }
}
}
int main () {
int arr[n][n];
cout<<"Enter elements of arrays : "<<endl;
int i,j;
for(int i=0;i<n;i++) {
for(int j=0;j<n;j++) {
cout<<"Coulom and row number of element is  ["<<i<<"]["<<j<<"] : ";
cin>>arr[i][j];
}
}
cout<<"In matrix form : "<<endl;
for(int i=0;i<n;i++) {
for(int  j=0;j<n;j++) {
cout<<arr[i][j]<<" ";
}
cout<<endl;}

transpose(arr);
cout<<"Transpose of above matrix is : "<<endl;
for(int i=0;i<n;i++) {
for(int j=0;j<n;j++) {
cout<<arr[i][j]<<" ";
}
cout<<endl;}

return 0;
}
```

RESULT:



```
Enter elements of arrays :
Coulom and row number of element is  [0][0] : 1
Coulom and row number of element is  [0][1] : 2
Coulom and row number of element is  [0][2] : 3
Coulom and row number of element is  [1][0] : 4
Coulom and row number of element is  [1][1] : 5
Coulom and row number of element is  [1][2] : 6
Coulom and row number of element is  [2][0] : 7
Coulom and row number of element is  [2][1] : 8
Coulom and row number of element is  [2][2] : 9
In matrix form :
1 2 3
4 5 6
7 8 9
Transpose of above matrix is :
1 4 7
2 5 8
3 6 9

--------------------------------
Process exited after 6.248 seconds with return value 0
Press any key to continue . . .
```

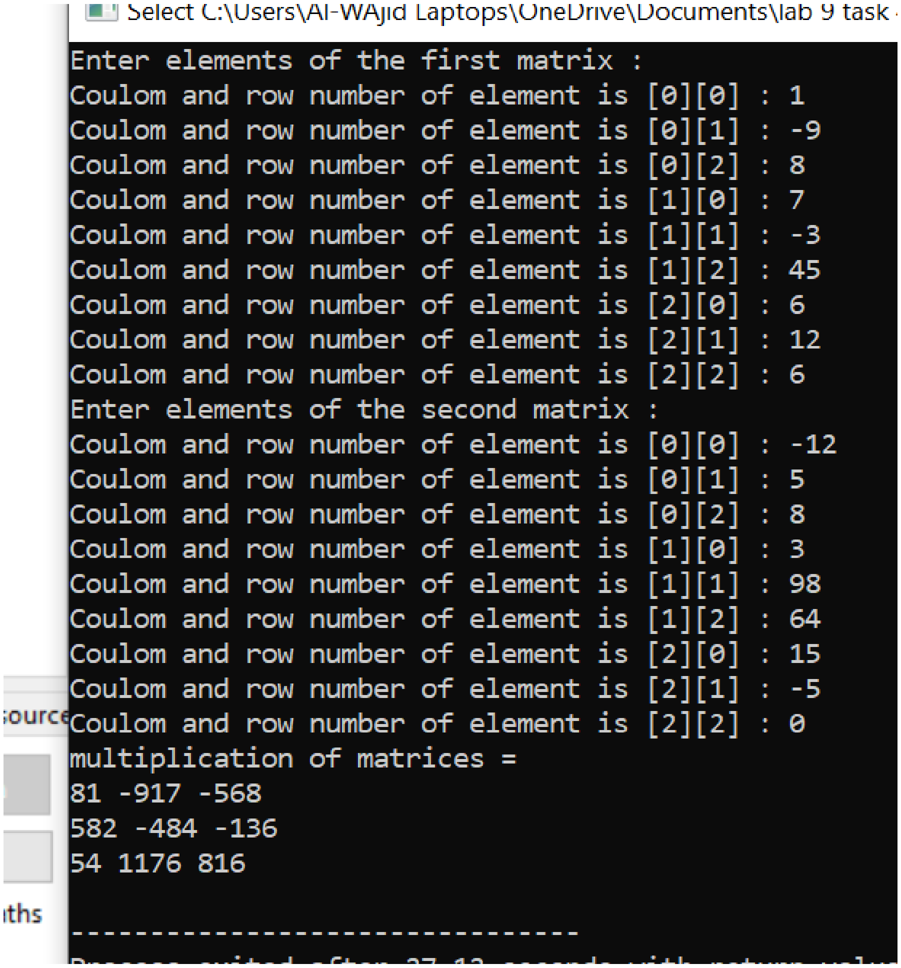
TASK 4:

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

CODE:

```
#include <bits/stdc++.h>

using namespace std;

const int size = 3;

// Function to multiply two 3x3 matrices

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 i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
        for(int k=0;k<size; k++) {
            result[i][j] =result[i][j]+ mat1[i][k] * mat2[k][j];
        }
    }
} }
int main() {
    int k, mat1[size][size];
    int mat2[size][size];
    int result[size][size];
    cout << "Enter elements of the first matrix : "<<endl;
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
        cout<<"Coulom and row number of element is ["<<i<<"]["<<j<<"] : ";
            cin >> mat1[i][j];
        }
    }

    cout << "Enter elements of the second matrix : "<<endl;
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
              cout<<"Coulom and row number of element is ["<<i<<"]["<<j<<"] : ";
            cin >> mat2[i][j];
        }
    }
    // Call the function to multiply matrices
    multiplymatrices(mat1, mat2, result);

    // Display the result
    cout << "multiplication of matrices = "<<endl;
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            cout << result[i][j] << " ";
        }
        cout << endl;
    }

    return 0;
}
```

RESULT:

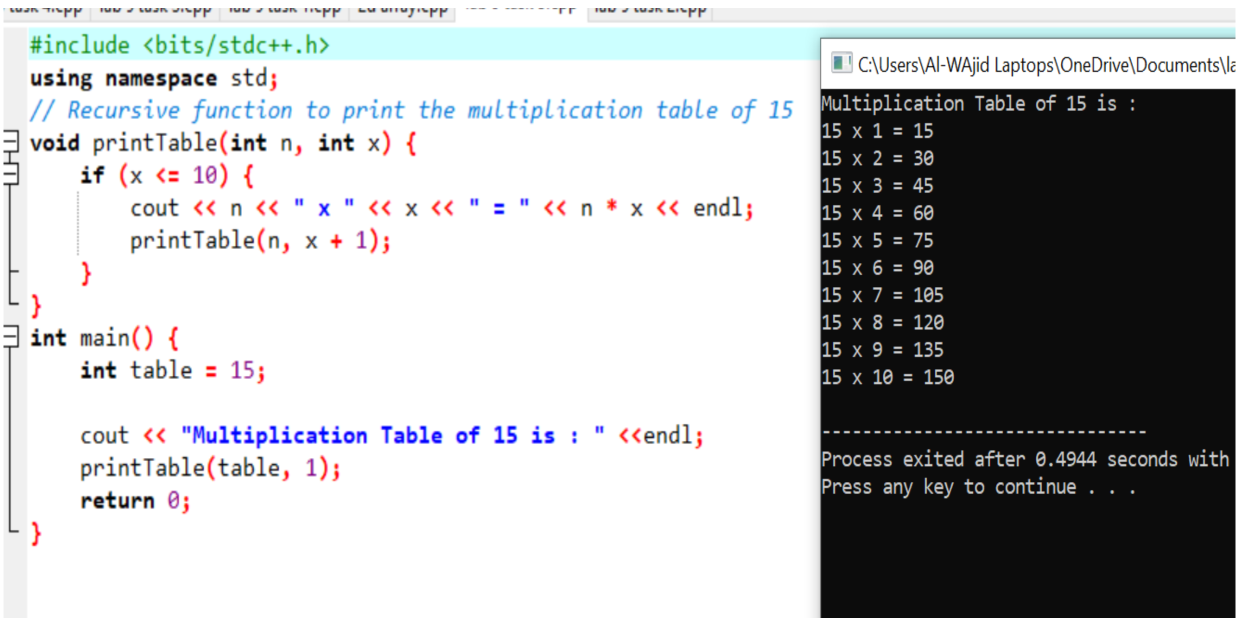

Select C:\Users\Al-WAjid Laptops\OneDrive\Documents\lab 9 task
ource
ths

```
Enter elements of the first matrix :
Coulom and row number of element is [0][0] : 1
Coulom and row number of element is [0][1] : -9
Coulom and row number of element is [0][2] : 8
Coulom and row number of element is [1][0] : 7
Coulom and row number of element is [1][1] : -3
Coulom and row number of element is [1][2] : 45
Coulom and row number of element is [2][0] : 6
Coulom and row number of element is [2][1] : 12
Coulom and row number of element is [2][2] : 6
Enter elements of the second matrix :
Coulom and row number of element is [0][0] : -12
Coulom and row number of element is [0][1] : 5
Coulom and row number of element is [0][2] : 8
Coulom and row number of element is [1][0] : 3
Coulom and row number of element is [1][1] : 98
Coulom and row number of element is [1][2] : 64
Coulom and row number of element is [2][0] : 15
Coulom and row number of element is [2][1] : -5
Coulom and row number of element is [2][2] : 0
multiplication of matrices =
81 -917 -568
582 -484 -136
54 1176 816


--------------------------------
Process exited after 27.12 seconds with return valu
```

TASK 5:

Print the multiplication table of 15 using recursion.

CODE:



```
#include <bits/stdc++.h>
using namespace std;
// Recursive function to print the multiplication table of 15
void printTable(int n, int x) {
    if (x <= 10) {
        cout << n << " x " << x << " = " << n * x << endl;
        printTable(n, x + 1);
    }
}
int main() {
    int table = 15;

    cout << "Multiplication Table of 15 is : " <<endl;
    printTable(table, 1);
    return 0;
}
```

```
Multiplication Table of 15 is :
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 0.4944 seconds with
Press any key to continue . . .
```

HOME TASK 1:

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

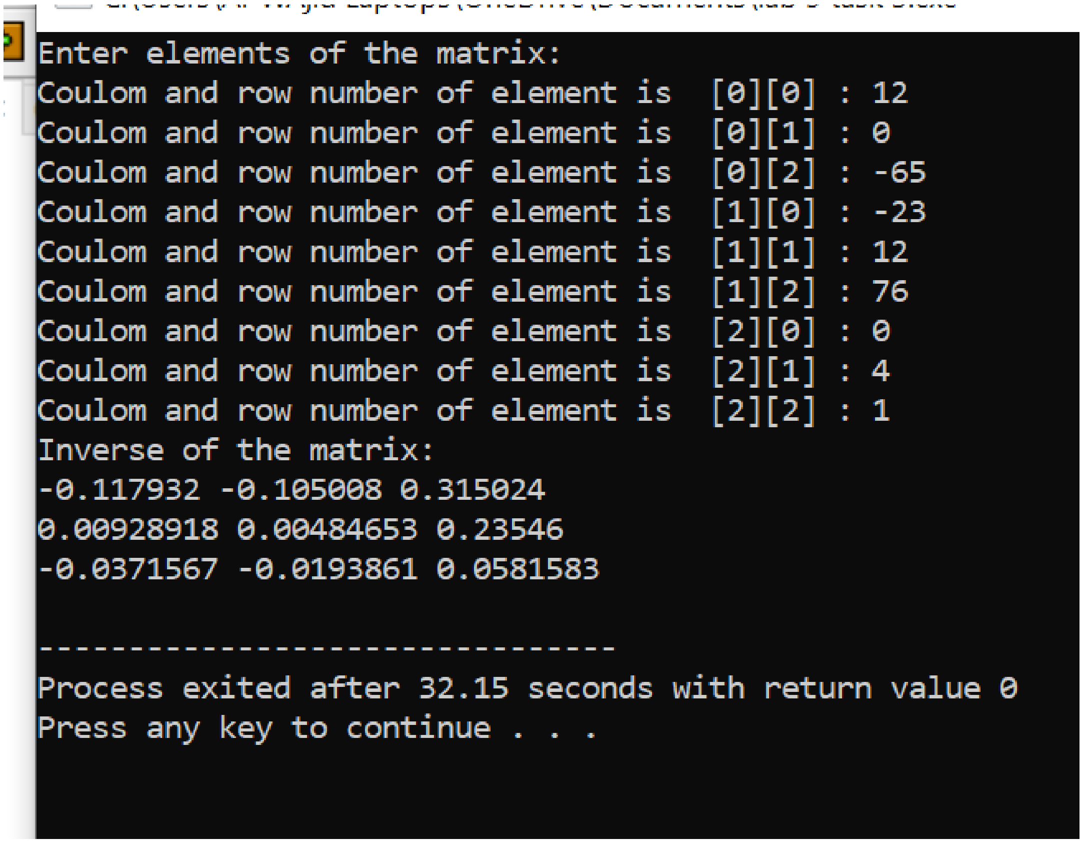
CODE:

```
#include<bits/stdc++.h>

using namespace std;

double Determinant(int 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]);

}

void Adjoint(int mat[3][3], int adj[3][3]) {

for (int i=0; i<3; i++) {

for (int j=0; j<3; j++) {

adj[i][j] = (mat[(j+1)%3][(i + 1)%3]* mat[(j+2)%3][(i+2)%3]) -

(mat[(j+1)%3][(i+2)%3]*mat[(j+2)%3][(i+1)%3]);

}

}

}
```

```
void Inverse(int mat[3][3], double inv[3][3]) {
double det = Determinant(mat);
if (det==0) {
cout<<"Inverse does not exist (Matrix is singular)!"<<endl;
return;
}
int adj[3][3];
Adjoint(mat,adj);
for (int i=0; i<3; i++) {
for (int j=0; j<3; j++) {
inv[i][j]=adj[i][j]/det;
}
}
}
int main() {
int mat[3][3];
cout << "Enter elements of the matrix:"<<endl;
for (int i=0; i<3; i++) {
for (int j=0; j<3; j++) {
cout<<"Coulom and row number of element is  ["<<i<<"]["<<j<<"] : ";
cin>>mat[i][j];
}
}
double inv[3][3];
Inverse(mat, inv);
if (Determinant(mat) != 0) {
cout << "Inverse of the matrix:"<<endl;
for (int i=0; i<3; i++) {
for (int j=0; j<3;j++) {
```

```
cout<<inv[i][j] << " ";

}

cout<<endl;

}

}

return 0;

}
```

RESULT:



```
Enter elements of the matrix:
Coulom and row number of element is  [0][0] : 12
Coulom and row number of element is  [0][1] : 0
Coulom and row number of element is  [0][2] : -65
Coulom and row number of element is  [1][0] : -23
Coulom and row number of element is  [1][1] : 12
Coulom and row number of element is  [1][2] : 76
Coulom and row number of element is  [2][0] : 0
Coulom and row number of element is  [2][1] : 4
Coulom and row number of element is  [2][2] : 1
Inverse of the matrix:
-0.117932 -0.105008 0.315024
0.00928918 0.00484653 0.23546
-0.0371567 -0.0193861 0.0581583


--------------------------------
Process exited after 32.15 seconds with return value 0
Press any key to continue . . .
```