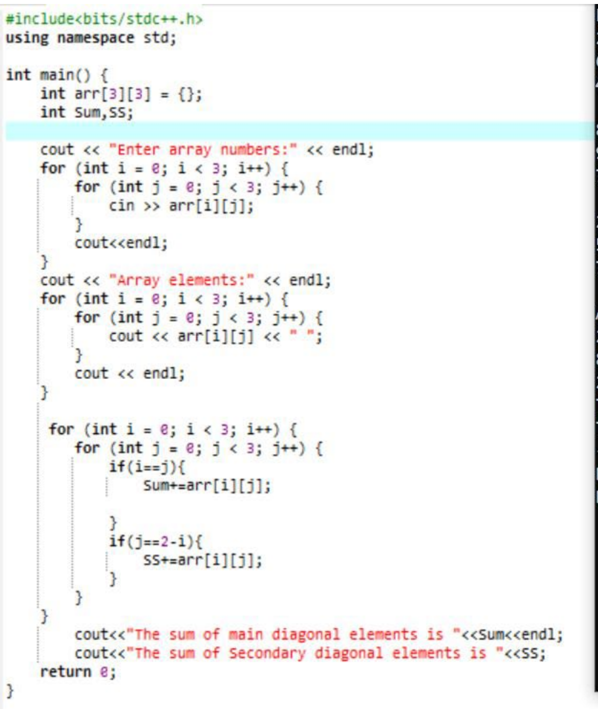
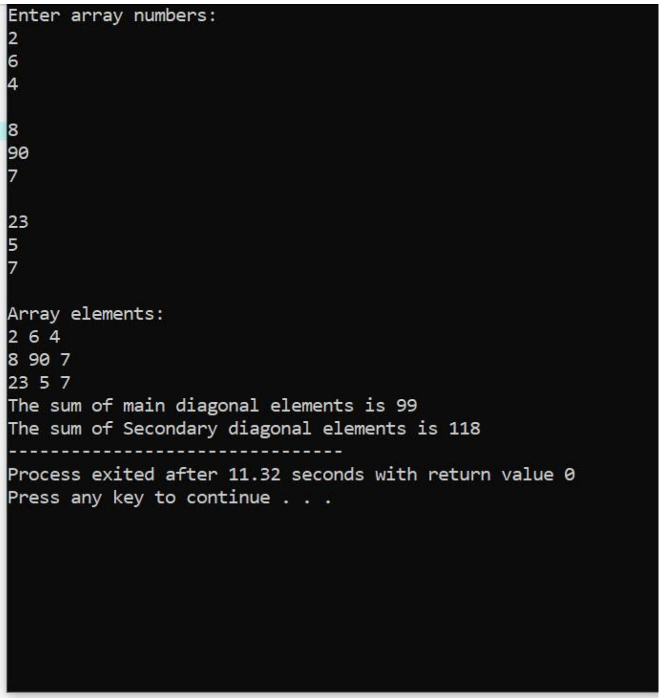
Make 2D Array in C++ and print left diagonal and right diagonal sum of a 3x3 matrix.



```
#include<bits/stdc++.h>
using namespace std;

int main() {
    int arr[3][3] = {};
    int Sum,SS;

    cout << "Enter array numbers:" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cin >> arr[i][j];
        }
        cout<<endl;
    }
    cout << "Array elements:" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << arr[i][j] << " ";
        }
        cout << endl;
    }

     for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if(i==j){
                Sum+=arr[i][j];

            }
            if(j==2-i){
                SS+=arr[i][j];
            }
        }
    }
        cout<<"The sum of main diagonal elements is "<<Sum<<endl;
        cout<<"The sum of Secondary diagonal elements is "<<SS;
    return 0;
}
```

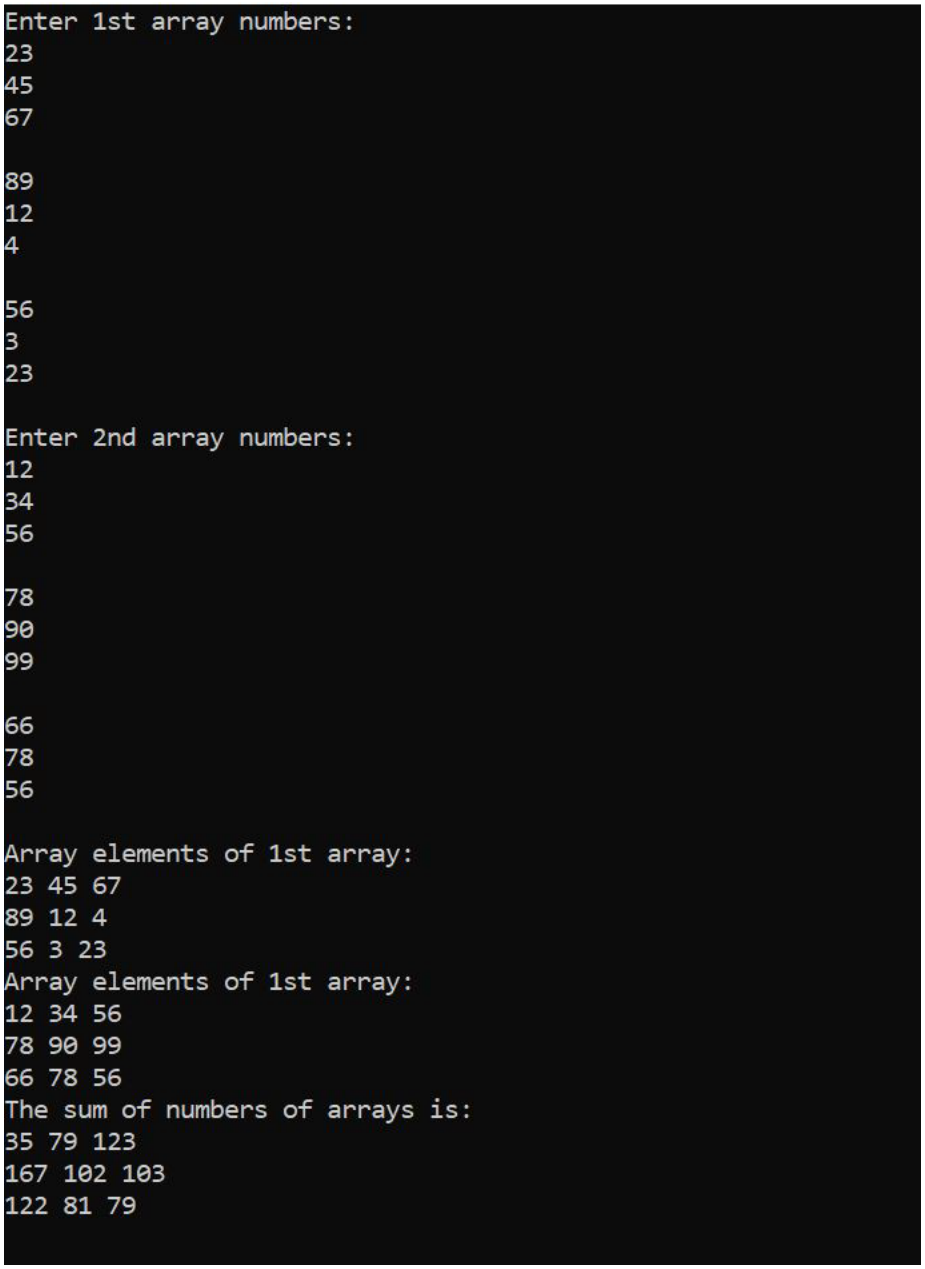


```
Enter array numbers:
2
6
4

8
90
7

23
5
7

Array elements:
2 6 4
8 90 7
23 5 7
The sum of main diagonal elements is 99
The sum of Secondary diagonal elements is 118
--------------------------------
Process exited after 11.32 seconds with return value 0
Press any key to continue . . .
```

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

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

using namespace std;

int main(){

               int arr[3][3] = {};

                              int arr1[3][3] = {};

   int Sum,SS;


   cout << "Enter 1st array numbers:" << endl;

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

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

         cin >> arr[i][j];

      }

      cout<<endl;

   }

   cout << "Enter 2nd array numbers:" << endl;

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

      for (int j = 0; j < 3; j++) {
            cin >> arr1[i][j];
        }
        cout<<endl;
    }
    cout << "Array elements of 1st array:" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << arr[i][j] << " ";
        }
        cout << endl;
    }
    cout << "Array elements of 1st array:" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << arr1[i][j] << " ";
        }
        cout << endl;
    }
    cout<<"The sum of numbers of arrays is: "<<endl;
    for(int i=0;i<3;i++){
                  for(int j=0;j<3;j++){
                                    Sum = arr[i][j]+arr1[i][j];
                                    cout<<Sum<<" ";
                                    }
                                    cout<<endl;
                  }
                  return 0;
}
```

Result:

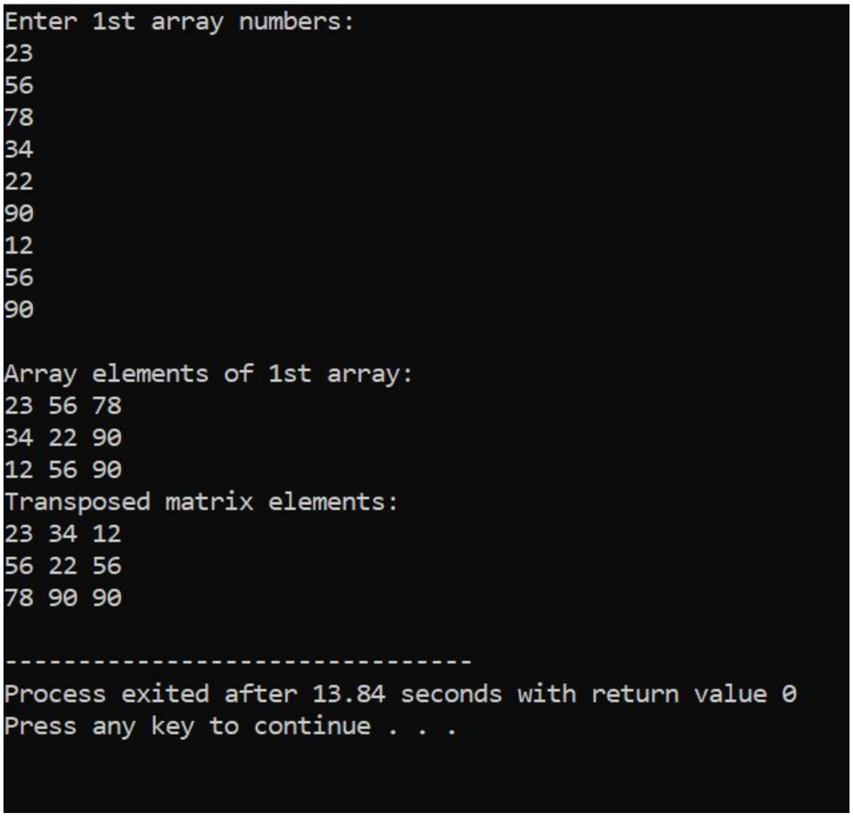


```
Enter 1st array numbers:
23
45
67

89
12
4

56
3
23

Enter 2nd array numbers:
12
34
56

78
90
99

66
78
56

Array elements of 1st array:
23 45 67
89 12 4
56 3 23
Array elements of 1st array:
12 34 56
78 90 99
66 78 56
The sum of numbers of arrays is:
35 79 123
167 102 103
122 81 79
```

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

```
#include<bits/stdc++.h>
using namespace std;
int main(){
                int arr [3][3];
                int transpose [3][3];
                cout << "Enter 1st array numbers:" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cin >> arr[i][j];
        }
    }
        cout<<endl;
         cout << "Array elements of 1st array:" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << arr[i][j] << " ";
        }
        cout << endl;
    }
     for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
           transpose[i][j]=arr[j][i];
        }

    }
     cout << "Transposed matrix elements:" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << transpose[i][j] << " ";
        }
        cout << endl;
    }
    return 0;

}
```

Result:



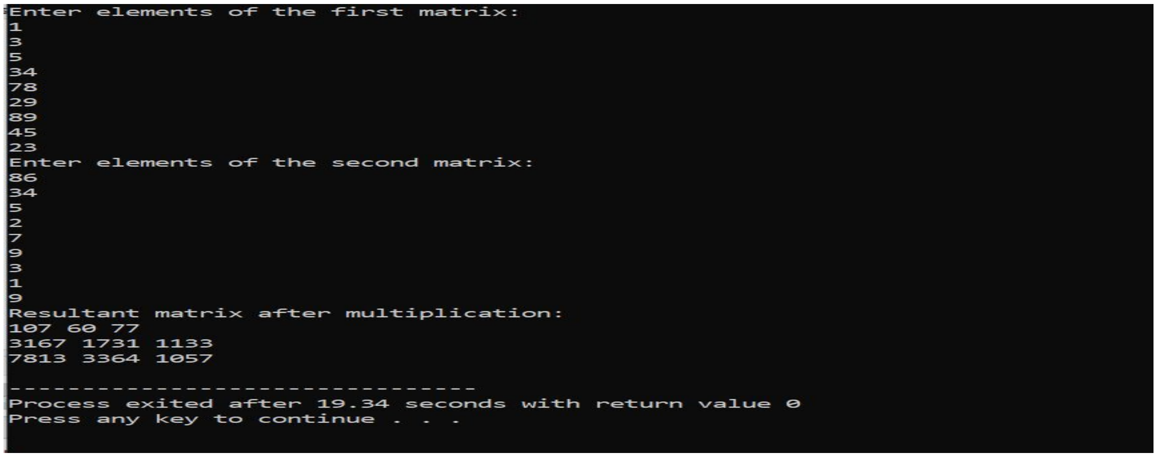
```
Enter 1st array numbers:
23
56
78
34
22
90
12
56
90

Array elements of 1st array:
23 56 78
34 22 90
12 56 90
Transposed matrix elements:
23 34 12
56 22 56
78 90 90

--------------------------------
Process exited after 13.84 seconds with return value 0
Press any key to continue . . .
```

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

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

using namespace std;


int main() {

                int N=3;

   int mat1[N][N], mat2[N][N], 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++) {

           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++) {
            cin >> mat2[i][j];
        }
    }
     for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            result[i][j] = 0;
            for (int k = 0; k < N; k++) {
                result[i][j] += mat1[i][k] * mat2[k][j];
            }
        }
    }
    cout << "Resultant matrix after multiplication:" << endl;
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            cout << result[i][j] << " ";
        }
        cout << endl;
    }

    return 0;
}
```

Result:

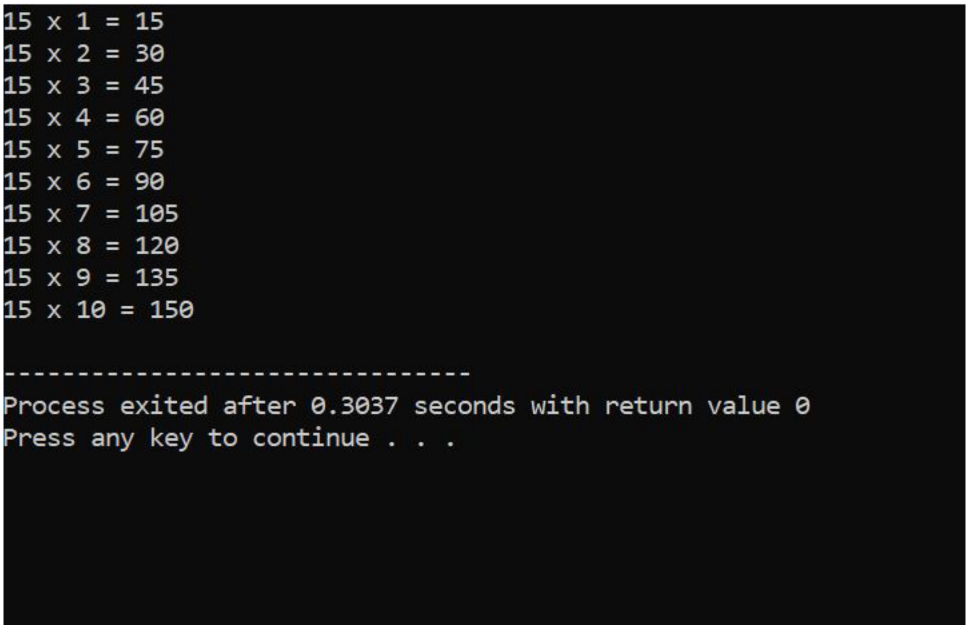


```
Enter elements of the first matrix:
1
3
5
34
78
29
89
45
23
Enter elements of the second matrix:
86
34
5
2
7
9
3
1
9
Resultant matrix after multiplication:
107 60 77
3167 1731 1133
7813 3364 1057

--------------------------------
Process exited after 19.34 seconds with return value 0
Press any key to continue . . .
```

Print the multiplication table of 15 using recursion.

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

void Table(int num, int mul) {
    if (mul > 10) {
                return;
                }
    cout << num << " x " << mul << " = " << num * mul << endl;
    Table(num, mul + 1);
}

int main() {
                int num=15,mul=1;
    Table(num,mul);
    return 0;
}
```

Result:

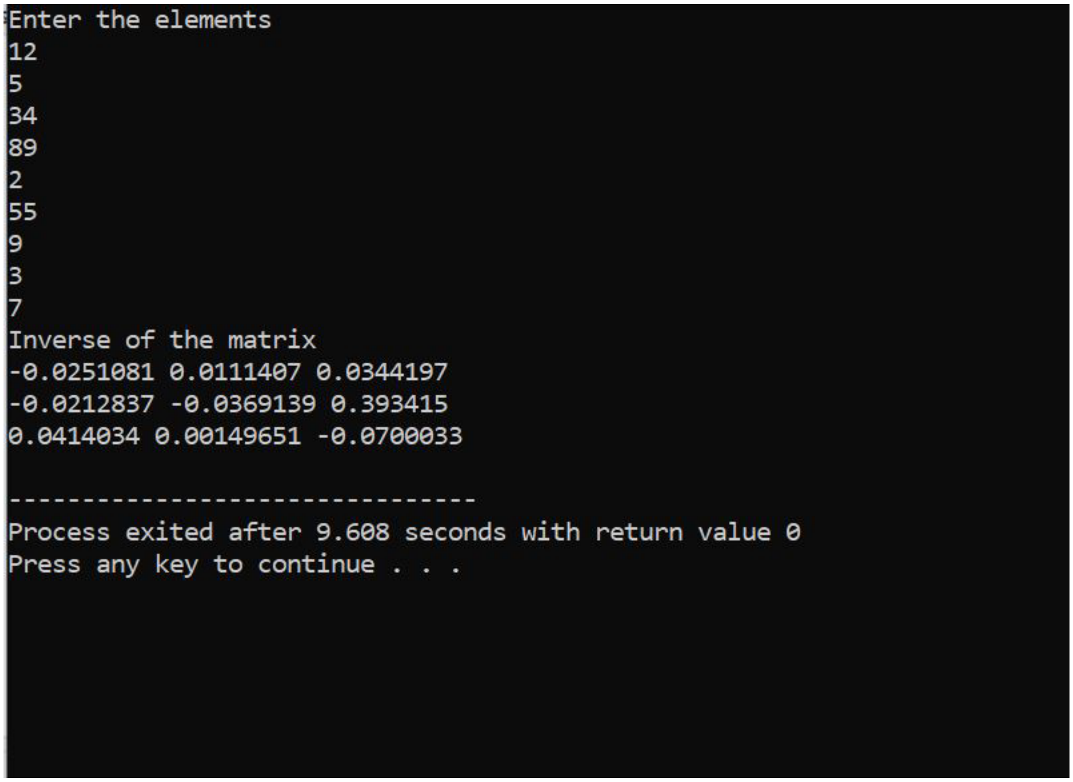


```
15 x 1 = 15
15 x 2 = 30
15 x 3 = 45
15 x 4 = 60
15 x 5 = 75
15 x 6 = 90
15 x 7 = 105
15 x 8 = 120
15 x 9 = 135
15 x 10 = 150

--------------------------------
Process exited after 0.3037 seconds with return value 0
Press any key to continue . . .
```

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

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


using namespace std;
double det(double mat[3][3]) {
    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 adjoint(double mat[3][3], double adj[3][3]) {
    adj[0][0] = mat[1][1] * mat[2][2] - mat[2][1] * mat[1][2];
    adj[0][1] = mat[0][2] * mat[2][1] - mat[2][2] * mat[0][1];
    adj[0][2] = mat[0][1] * mat[1][2] - mat[1][1] * mat[0][2];

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

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

void inverse(double mat[3][3], double inv[3][3]) {
    double deter = det(mat);

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

    double 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] / deter;
        }
    }
}

int main() {
    double matrix[3][3];

    cout << "Enter the elements " << endl;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cin >> matrix[i][j];
        }
    }

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

    cout << "Inverse of the matrix " << endl;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << inverseMatrix[i][j] << " ";
        }
        cout << endl;
    }

    return 0;
}
```

Result:



```
Enter the elements
12
5
34
89
2
55
9
3
7
Inverse of the matrix
-0.0251081 0.0111407 0.0344197
-0.0212837 -0.0369139 0.393415
0.0414034 0.00149651 -0.0700033

--------------------------------
Process exited after 9.608 seconds with return value 0
Press any key to continue . . .
```