Fundamentals of Programming

Lab Manual 9, Lab tasks 1-5

Me-15

Name:	Faizan Ahmad
Qalam ID:	476602
Section:	A

Task 1:

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

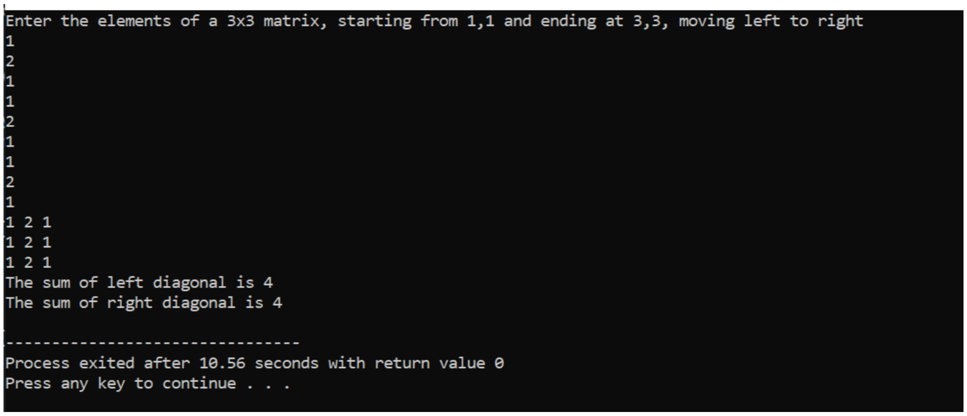
Code:

```
#include <iostream>
using namespace std;
int main(){

//Task 1
//Make 2D Array in C++ and print left diagonal and right diagonal sum of a 3x3 matrix



int sum_right=0, sum_left=0, x, y ;
int matrix[3][3] = {};
cout<<"Enter the elements of a 3x3 matrix, starting from 1,1 and ending at 3,3, moving left to right "<<endl;

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


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

cin>>matrix [i][j];
}
}

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++ ) {

y = i;
for ( int j = 0; j<3; j++ ) {
x = j;
if ( i == j ) {
sum_left = sum_left + matrix[i][j];
}
if ( x + y == 2 ) {
sum_right = sum_right + matrix[i][j];
}

}

}

cout<<"The sum of left diagonal is "<<sum_left<<endl;
```

```
cout<<"The sum of right diagonal is "<<sum_right<<endl;


return 0;

}
```

Execute:



```
Enter the elements of a 3x3 matrix, starting from 1,1 and ending at 3,3, moving left to right
1
2
1
1
2
1
1
2
1
1 2 1
1 2 1
1 2 1
The sum of left diagonal is 4
The sum of right diagonal is 4

--------------------------------
Process exited after 10.56 seconds with return value 0
Press any key to continue . . .
```

Task 2:

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

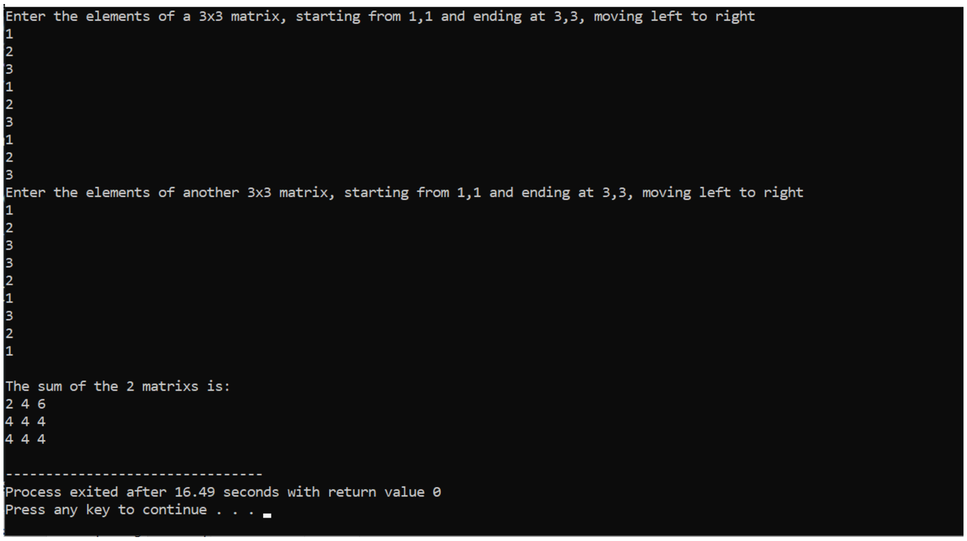
Code:

```
#include<iostream>

using namespace std;



void matrix_sum ( int matrix1 [3][3] , int matrix2 [3][3], int sum[3][3]  ) {
```

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

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

sum[i][j] = matrix1[i][j] + matrix2[i][j];
}
}

}

int main () {

int sum[3][3] = {};
int matrix1[3][3] = {};
int matrix2[3][3] = {};
cout<<"Enter the elements of a 3x3 matrix, starting from 1,1 and ending at 3,3, moving left to right "<<endl;

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

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

cin>>matrix1 [i][j];
}

}


cout<<"Enter the elements of another 3x3 matrix, starting from 1,1 and ending at 3,3, moving left to right "<<endl;


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



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


cin>>matrix2 [i][j];

}

}



cout<<endl<<"The sum of the 2 matrixs is: "<<endl;


matrix_sum ( matrix1, matrix2, sum );


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

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


cout<<sum[i][j]<<" ";

}

cout<<endl;

}
```

```
return 0;

}
```

Execute:



```
Enter the elements of a 3x3 matrix, starting from 1,1 and ending at 3,3, moving left to right
1
2
3
1
2
3
1
2
3
Enter the elements of another 3x3 matrix, starting from 1,1 and ending at 3,3, moving left to right
1
2
3
3
2
1
3
2
1

The sum of the 2 matrixs is:
2 4 6
4 4 4
4 4 4

--------------------------------
Process exited after 16.49 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.

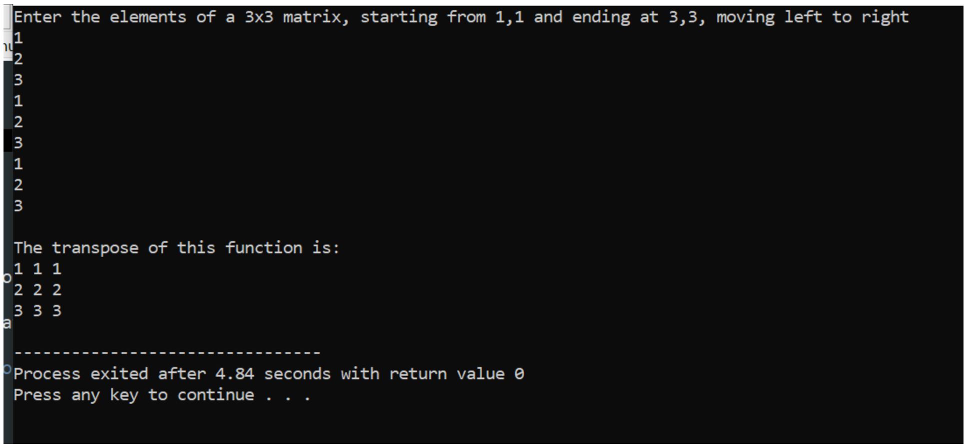
Code:

```
#include<iostream>

using namespace std;



void matrix_transpose ( int matrix [3][3] , int transpose[3][3]  ) {
```

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

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

transpose[i][j] = matrix[j][i];
}
}


}


int main () {


int transpose[3][3] = {};
int matrix[3][3] = {};

cout<<"Enter the elements of a 3x3 matrix, starting from 1,1 and ending at 3,3, moving left to right "<<endl;

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


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

```
cin>>matrix [i][j];
}
}



cout<<endl<<"The transpose of the matrix is: "<<endl;

matrix_transpose ( matrix, transpose );

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

cout<<transpose[i][j]<<" ";
}
cout<<endl;
}


return 0;
}
```

Execute:



```
Enter the elements of a 3x3 matrix, starting from 1,1 and ending at 3,3, moving left to right
1
2
3
1
2
3
1
2
3

The transpose of this function is:
1 1 1
2 2 2
3 3 3

--------------------------------
Process exited after 4.84 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<iostream>
using namespace std;


void matrix_product ( int matrix1[3][3], int matrix2[3][3], int product[3][3] ) {

for (int i=0; i<3; i++) {
for (int j=0; j<3; j++ ) {
int sum=0;
for ( int k=0; k<3; k++ ) {
```

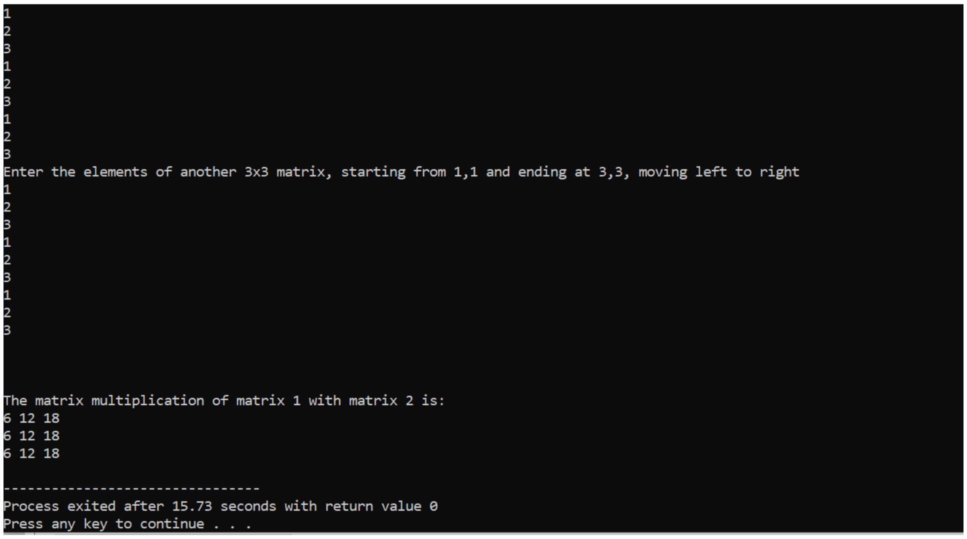
```
sum = sum + (matrix1[i][k] * matrix2[k][j]);



}
product[i][j] = sum;
}
cout<<endl;
}
}




int main() {




int multi[3][3] = {};
int matrix1[3][3] = {};
int matrix2[3][3] = {};
cout<<"Enter the elements of a 3x3 matrix, starting from 1,1 and ending at 3,3, moving left to
right "<<endl;

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


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

```
cin>>matrix1 [i][j];

}

}


cout<<"Enter the elements of another 3x3 matrix, starting from 1,1 and ending at 3,3, moving left to right "<<endl;


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



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


cin>>matrix2 [i][j];

}

}


matrix_product( matrix1, matrix2, multi);


cout<<"The matrix multiplication of matrix 1 with matrix 2 is: "<<endl;

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


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


cout<<multi[i][j]<<" ";

}

cout<<endl;

}
```

return 0;

}

Execute:



```
1
2
3
1
2
3
1
2
3
Enter the elements of another 3x3 matrix, starting from 1,1 and ending at 3,3, moving left to right
1
2
3
1
2
3
1
2
3



The matrix multiplication of matrix 1 with matrix 2 is:
6 12 18
6 12 18
6 12 18

--------------------------------
Process exited after 15.73 seconds with return value 0
Press any key to continue . . .
```

Task 5:

Print the multiplication table of 15 using recursion.

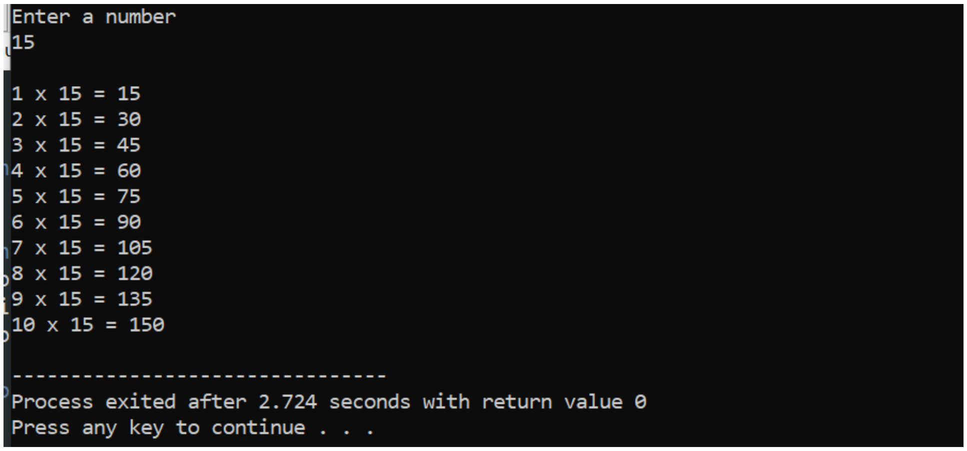
Code:

#include<iostream>

using namespace std;

```
void f_multiply (int number, int product) {
static int i=1;
if (i<=10) {
product = number * i;
cout<<i<<" x "<<number<<" = "<<product<<endl;
i++;

}
}



int main () {


int n , ans = 0;
cout<<"Enter a number "<<endl;
cin>>n;
cout<<endl;

for (int i=0; i<10; i++) {
f_multiply (n , ans);
}
return 0;

}
```

Execute:



```
Enter a number
15

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

--------------------------------
Process exited after 2.724 seconds with return value 0
Press any key to continue . . .
```

HOME TASKS:

Task:

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

Code:

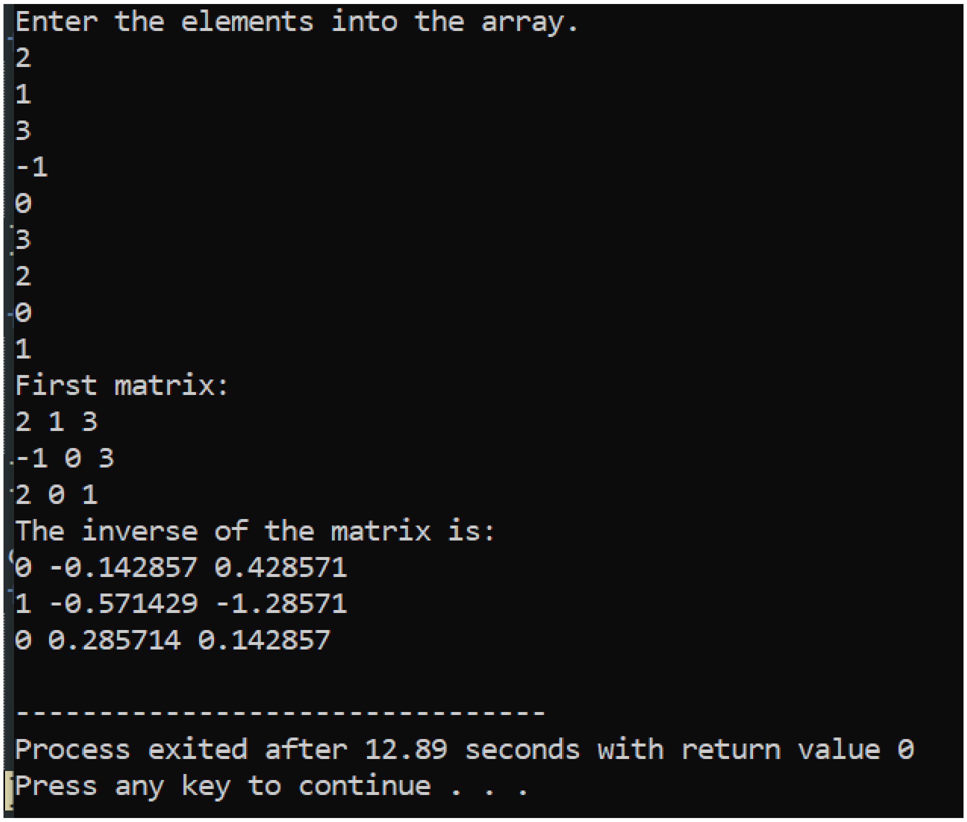
```
#include<iostream>

using namespace std;


void array(int matrix[3][3]) {

   cout << "Enter the elements into the array.\n";

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

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

         cin >> matrix[i][j];
```

```
        }
    }
}

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

int main() {

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

    array(matrix);
    cout << "First matrix: \n";
    array_display(matrix);

    float determinant = matrix[0][0] * (matrix[1][1] * matrix[2][2] - matrix[2][1] * matrix[1][2]) -
                        matrix[0][1] * (matrix[1][0] * matrix[2][2] - matrix[2][0] * matrix[1][2]) +
                        matrix[0][2] * (matrix[1][0] * matrix[2][1] - matrix[2][0] * matrix[1][1]);

    if (determinant == 0) {
        cout << "The matrix is singular, its inverse does not exist." << endl;
```

```
}
else {
    float adjoint[3][3], inverse[3][3];

    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            adjoint[i][j] = (matrix[(j + 1) % 3][(i + 1) % 3] * matrix[(j + 2) % 3][(i + 2) % 3] -
                            matrix[(j + 1) % 3][(i + 2) % 3] * matrix[(j + 2) % 3][(i + 1) % 3]);
        }
    }

    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            inverse[i][j] = adjoint[i][j] / determinant;
        }
    }

    cout << "The inverse of the matrix is:" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << inverse[i][j] << " ";
        }
        cout << endl;
    }
}

return 0;
```

}

Execute:



```
Enter the elements into the array.
2
1
3
-1
0
3
2
0
1
First matrix:
2 1 3
-1 0 3
2 0 1
The inverse of the matrix is:
0 -0.142857 0.428571
1 -0.571429 -1.28571
0 0.285714 0.142857

--------------------------------
Process exited after 12.89 seconds with return value 0
Press any key to continue . . .
```