CSS-114- FUNDAMENTALS OF PROGRAMMING

LAB MANUAL #9

LAB TASK

Course Instructor: Dr. Jawad Khan

Lab Instructor: Muhammad Affan

Student Name: ABDULLAH BIN KHORRAM

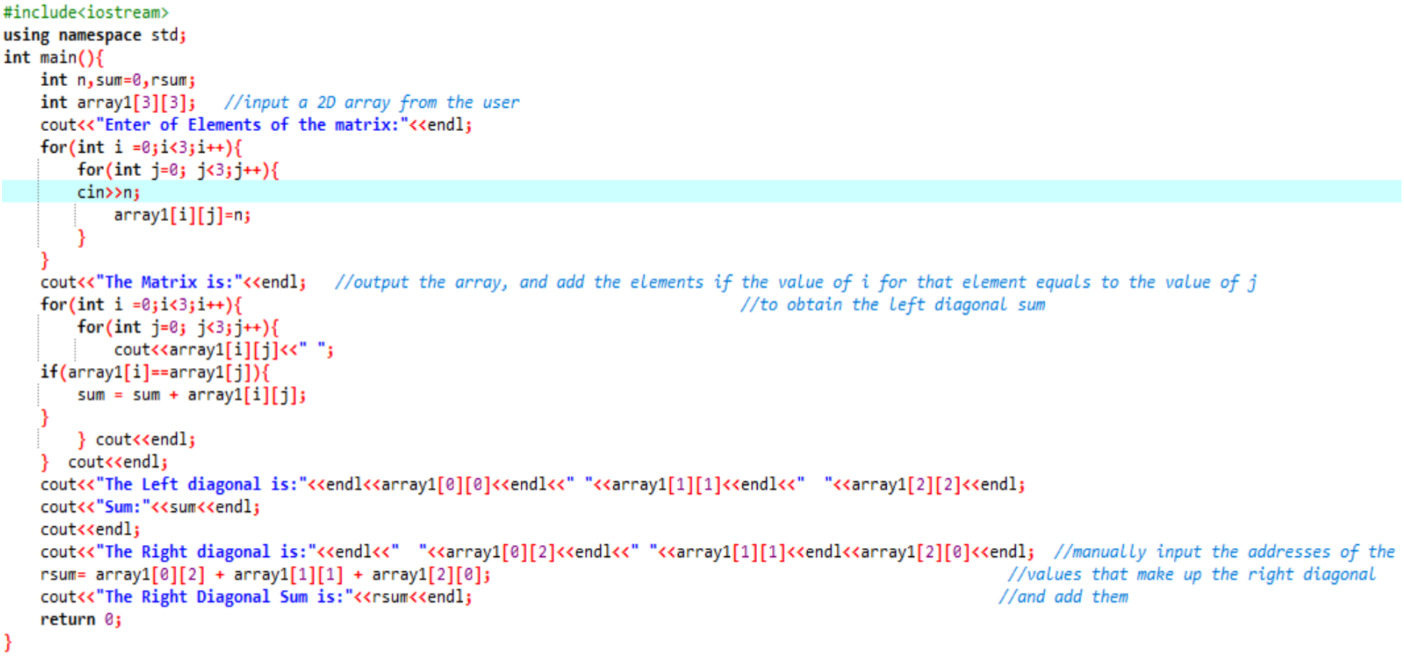
CMS ID: 466612

DATE: 6 DECEMBER 2023.

Lab Task:

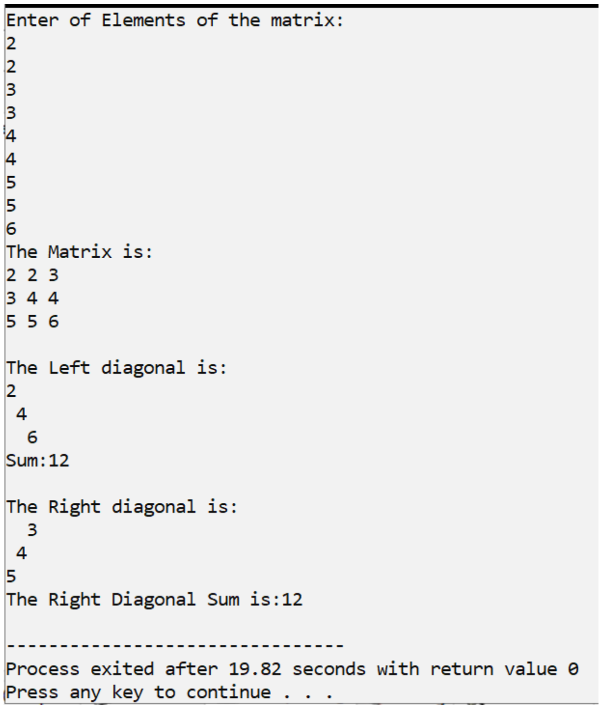
1. Make 2D Array in C++ and print left diagonal and right diagonal sum of a 3x3 matrix.

CODE:



```
#include<iostream>
using namespace std;
int main(){
    int n,sum=0,rsum;
    int array1[3][3];   //input a 2D array from the user
    cout<<"Enter of Elements of the matrix:"<<endl;
    for(int i =0;i<3;i++){
        for(int j=0; j<3;j++){
        cin>>n;
            array1[i][j]=n;
        }
    }
    cout<<"The Matrix is:"<<endl;   //output the array, and add the elements if the value of i for that element equals to the value of j
    for(int i =0;i<3;i++){                                               //to obtain the left diagonal sum
        for(int j=0; j<3;j++){
            cout<<array1[i][j]<<" ";
    if(array1[i]==array1[j]){
        sum = sum + array1[i][j];
    }
        } cout<<endl;
    }  cout<<endl;
    cout<<"The Left diagonal is:"<<endl<<array1[0][0]<<endl<<" "<<array1[1][1]<<endl<<"  "<<array1[2][2]<<endl;
    cout<<"Sum:"<<sum<<endl;
    cout<<endl;
    cout<<"The Right diagonal is:"<<endl<<"  "<<array1[0][2]<<endl<<" "<<array1[1][1]<<endl<<array1[2][0]<<endl;  //manually input the addresses of the
    rsum= array1[0][2] + array1[1][1] + array1[2][0];                                                           //values that make up the right diagonal
    cout<<"The Right Diagonal Sum is:"<<rsum<<endl;                                                            //and add them
    return 0;
}
```

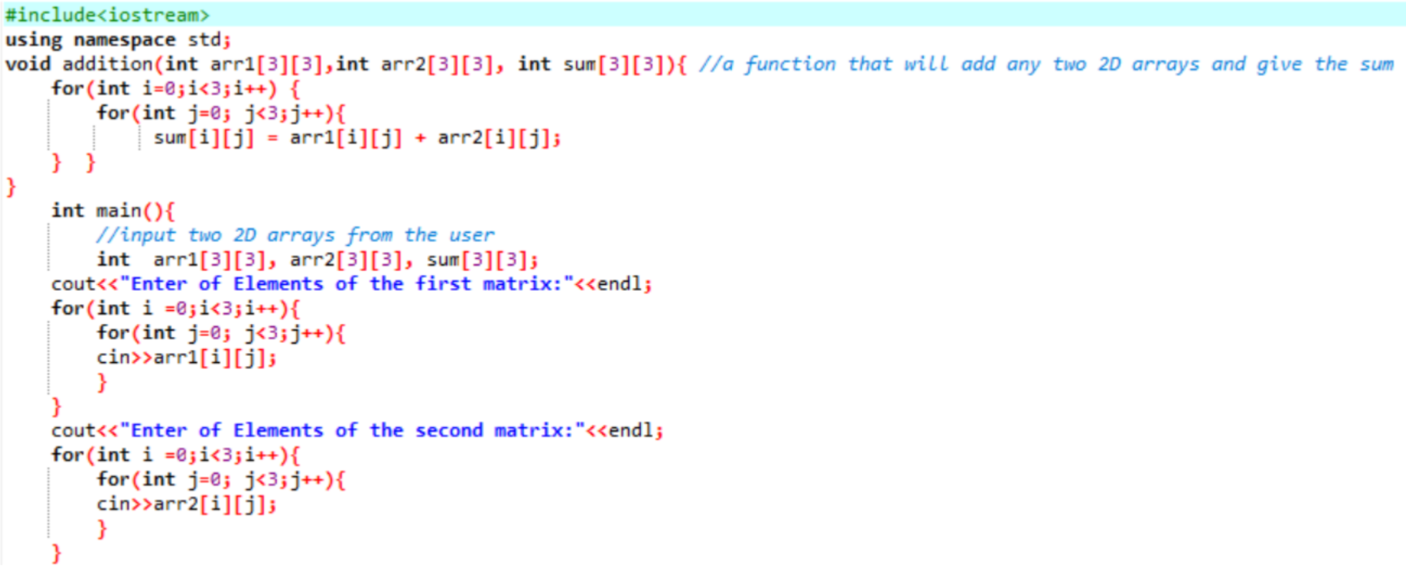
OUTPUT:



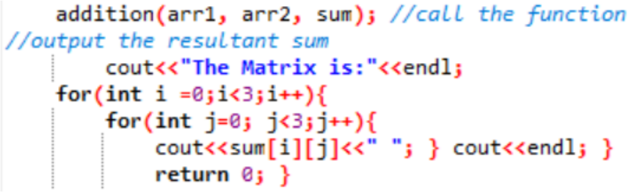
```
Enter of Elements of the matrix:
2
2
3
3
4
4
5
5
6
The Matrix is:
2 2 3
3 4 4
5 5 6

The Left diagonal is:
2
 4
  6
Sum:12

The Right diagonal is:
  3
 4
5
The Right Diagonal Sum is:12

--------------------------------
Process exited after 19.82 seconds with return value 0
Press any key to continue . . .
```

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

CODE:

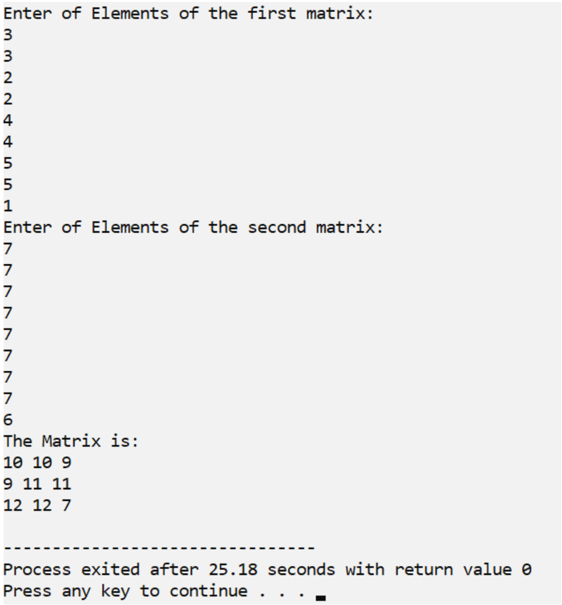


```cpp
#include<iostream>
using namespace std;
void addition(int arr1[3][3],int arr2[3][3], int sum[3][3]){ //a function that will add any two 2D arrays and give the sum
    for(int i=0;i<3;i++) {
        for(int j=0; j<3;j++){
            sum[i][j] = arr1[i][j] + arr2[i][j];
    }  }
}
    int main(){
        //input two 2D arrays from the user
        int  arr1[3][3], arr2[3][3], sum[3][3];
    cout<<"Enter of Elements of the first matrix:"<<endl;
    for(int i =0;i<3;i++){
        for(int j=0; j<3;j++){
        cin>>arr1[i][j];
        }
    }
    cout<<"Enter of Elements of the second matrix:"<<endl;
    for(int i =0;i<3;i++){
        for(int j=0; j<3;j++){
        cin>>arr2[i][j];
        }
    }
```



```cpp
     addition(arr1, arr2, sum); //call the function
//output the resultant sum
         cout<<"The Matrix is:"<<endl;
     for(int i =0;i<3;i++){
         for(int j=0; j<3;j++){
             cout<<sum[i][j]<<" "; } cout<<endl; }
             return 0; }
```

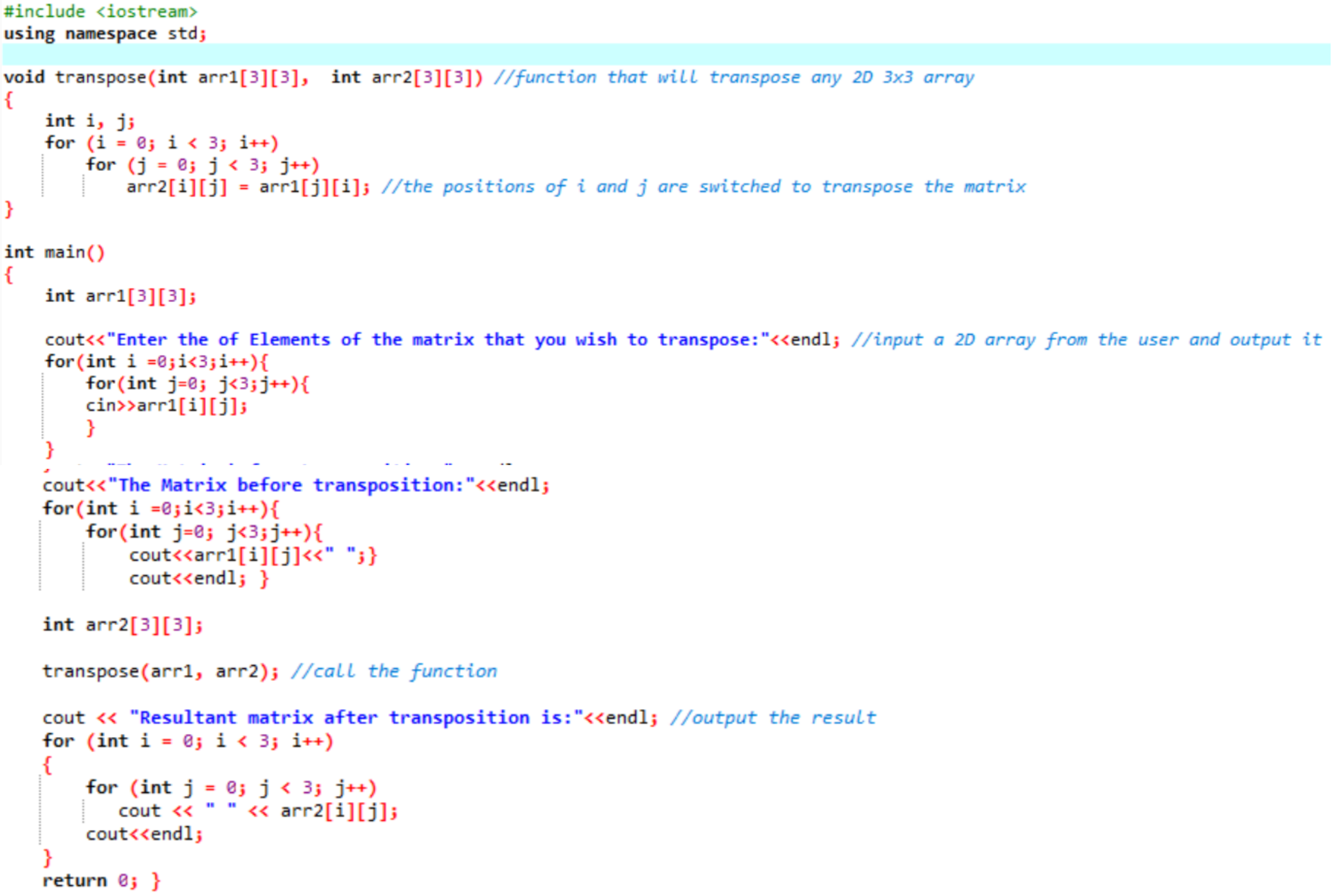
OUTPUT:



```
Enter of Elements of the first matrix:
3
3
2
2
4
4
5
5
1
Enter of Elements of the second matrix:
7
7
7
7
7
7
7
7
6
The Matrix is:
10 10 9
9 11 11
12 12 7

--------------------------------
Process exited after 25.18 seconds with return value 0
Press any key to continue . . .
```

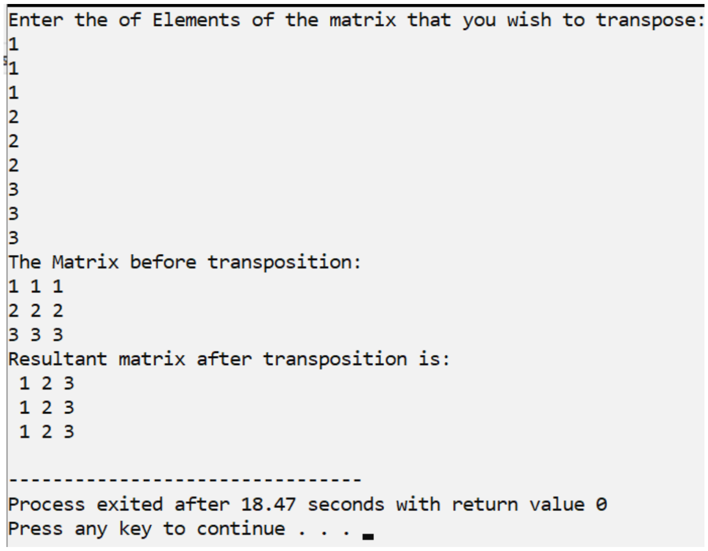
3. Using 2D arrays in C++, take transpose of a 3x3 matrix. Make a transpose function.

CODE:



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

void transpose(int arr1[3][3],  int arr2[3][3]) //function that will transpose any 2D 3x3 array
{
    int i, j;
    for (i = 0; i < 3; i++)
        for (j = 0; j < 3; j++)
            arr2[i][j] = arr1[j][i]; //the positions of i and j are switched to transpose the matrix
}

int main()
{
    int arr1[3][3];

    cout<<"Enter the of Elements of the matrix that you wish to transpose:"<<endl; //input a 2D array from the user and output it
    for(int i =0;i<3;i++){
        for(int j=0; j<3;j++){
        cin>>arr1[i][j];
        }
    }
    cout<<"The Matrix before transposition:"<<endl;
    for(int i =0;i<3;i++){
        for(int j=0; j<3;j++){
            cout<<arr1[i][j]<<" ";}
            cout<<endl; }

    int arr2[3][3];

    transpose(arr1, arr2); //call the function

    cout << "Resultant matrix after transposition is:"<<endl; //output the result
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
           cout << " " << arr2[i][j];
        cout<<endl;
    }
    return 0; }
```

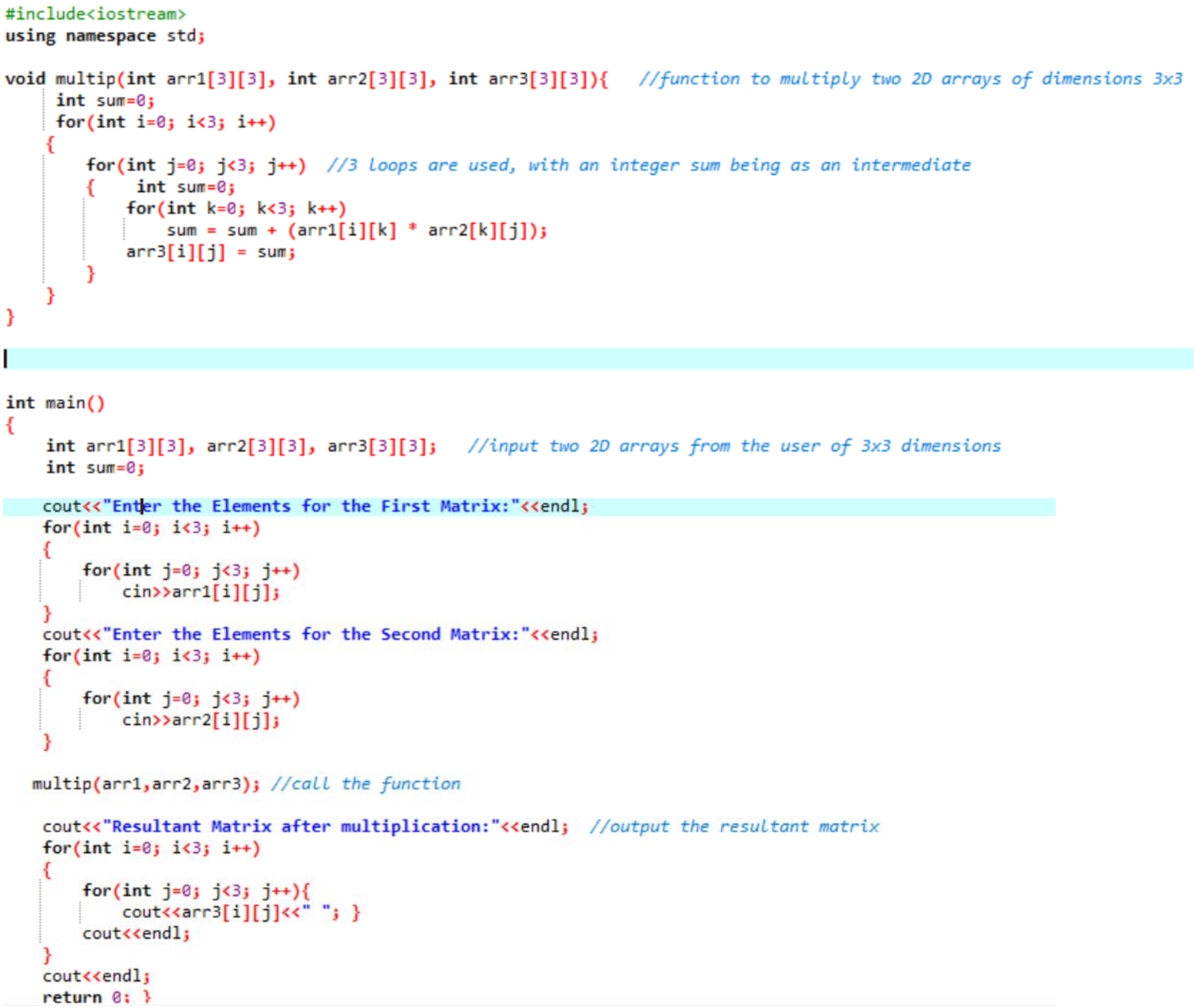
OUTPUT:



```
Enter the of Elements of the matrix that you wish to transpose:
1
1
1
2
2
2
3
3
3
The Matrix before transposition:
1 1 1
2 2 2
3 3 3
Resultant matrix after transposition is:
 1 2 3
 1 2 3
 1 2 3

--------------------------------
Process exited after 18.47 seconds with return value 0
Press any key to continue . . .
```

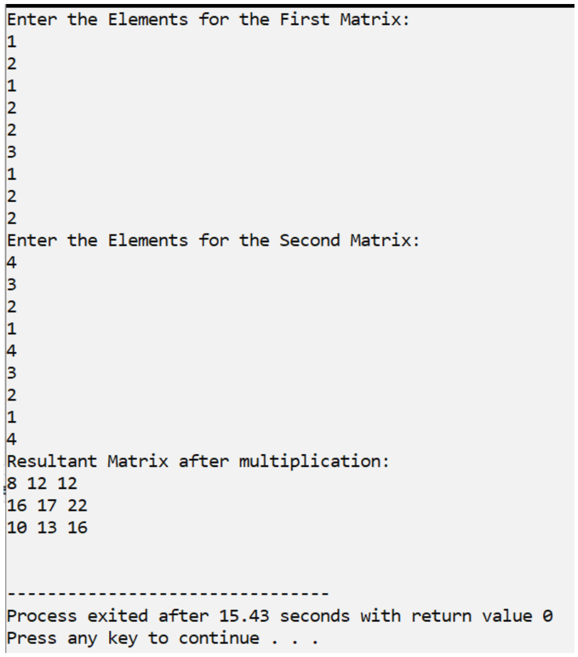
4. Using 2D arrays in C++, implement 3x3 matrix multiplication. Make a function.

CODE:



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

void multip(int arr1[3][3], int arr2[3][3], int arr3[3][3]){   //function to multiply two 2D arrays of dimensions 3x3
     int sum=0;
     for(int i=0; i<3; i++)
    {
        for(int j=0; j<3; j++)  //3 loops are used, with an integer sum being as an intermediate
        {    int sum=0;
            for(int k=0; k<3; k++)
                sum = sum + (arr1[i][k] * arr2[k][j]);
            arr3[i][j] = sum;
        }
    }
}


int main()
{
    int arr1[3][3], arr2[3][3], arr3[3][3];   //input two 2D arrays from the user of 3x3 dimensions
    int sum=0;

    cout<<"Enter the Elements for the First Matrix:"<<endl;
    for(int i=0; i<3; i++)
    {
        for(int j=0; j<3; j++)
            cin>>arr1[i][j];
    }
    cout<<"Enter the Elements for the Second Matrix:"<<endl;
    for(int i=0; i<3; i++)
    {
        for(int j=0; j<3; j++)
            cin>>arr2[i][j];
    }

   multip(arr1,arr2,arr3); //call the function

    cout<<"Resultant Matrix after multiplication:"<<endl;  //output the resultant matrix
    for(int i=0; i<3; i++)
    {
        for(int j=0; j<3; j++){
            cout<<arr3[i][j]<<" "; }
        cout<<endl;
    }
    cout<<endl;
    return 0; }
```

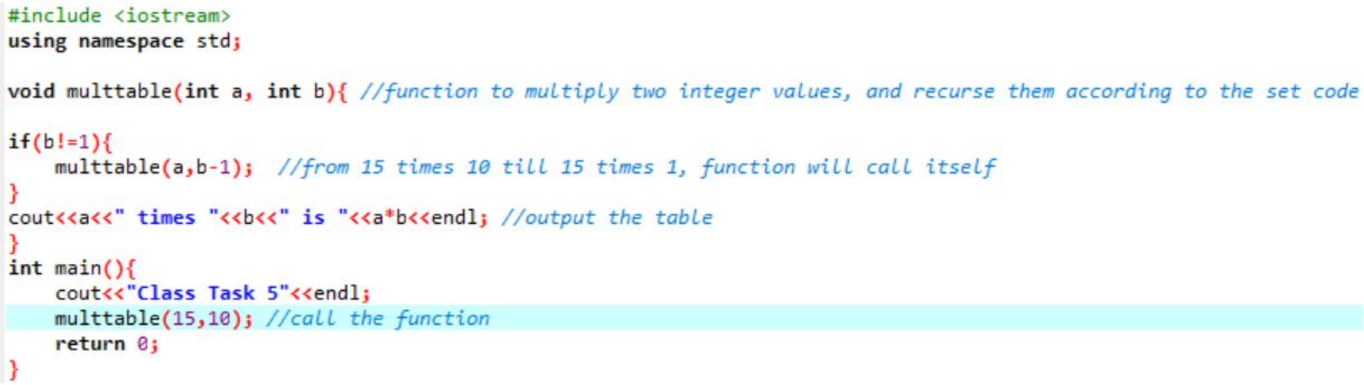
OUTPUT:



```
Enter the Elements for the First Matrix:
1
2
1
2
2
3
1
2
2
Enter the Elements for the Second Matrix:
4
3
2
1
4
3
2
1
4
Resultant Matrix after multiplication:
8 12 12
16 17 22
10 13 16


--------------------------------
Process exited after 15.43 seconds with return value 0
Press any key to continue . . .
```

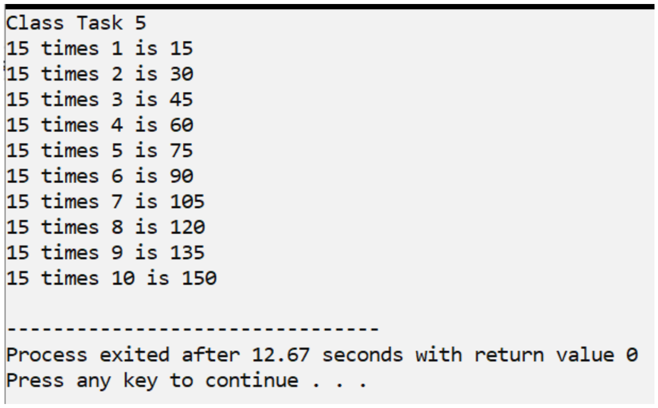
5. Print the multiplication table of 15 using recursion.

CODE:



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

void multtable(int a, int b){ //function to multiply two integer values, and recurse them according to the set code

if(b!=1){
    multtable(a,b-1);  //from 15 times 10 till 15 times 1, function will call itself
}
cout<<a<<" times "<<b<<" is "<<a*b<<endl; //output the table
}
int main(){
    cout<<"Class Task 5"<<endl;
    multtable(15,10); //call the function
    return 0;
}
```

OUTPUT:



```
Class Task 5
15 times 1 is 15
15 times 2 is 30
15 times 3 is 45
15 times 4 is 60
15 times 5 is 75
15 times 6 is 90
15 times 7 is 105
15 times 8 is 120
15 times 9 is 135
15 times 10 is 150

--------------------------------
Process exited after 12.67 seconds with return value 0
Press any key to continue . . .
```