NUST SCHOOL OF MECHANICAL &MANUFACTURING ENGINEERING

Lab Manual:9

Name: Ramzan Sameer

Batch: ME-15

Section: A

Qalam Id: 464899

Course: FOP

Course Instructor: Dr. Jawad

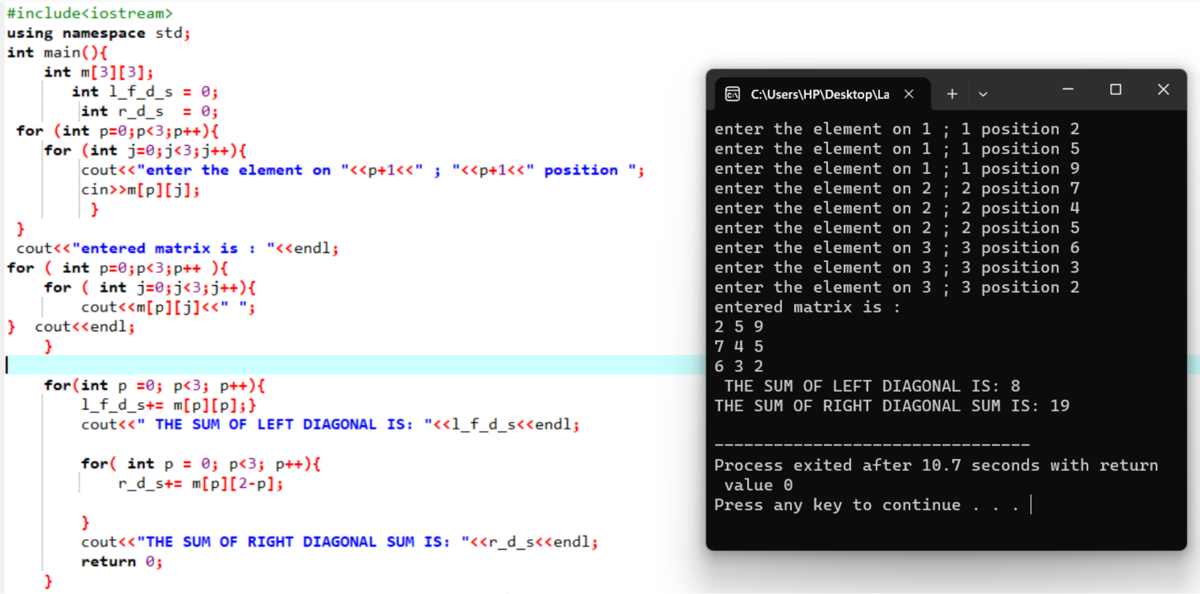
Lab Instructor: Sir Affan.

Date: 11 /12 /2023

SMME ♡

Lab Task:

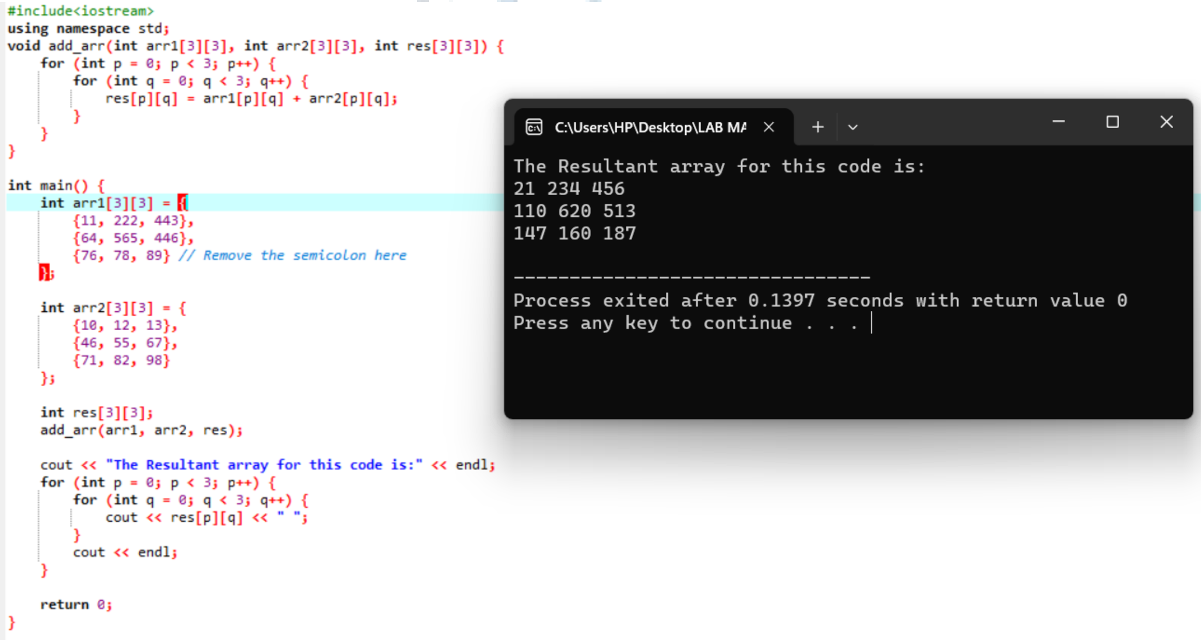
1. Make a 2D Array in C++ and print the left diagonal and right diagonal sum of a 3x3 matrix.



```
#include<iostream>
using namespace std;
int main(){
    int m[3][3];
       int l_f_d_s = 0;
        int r_d_s  = 0;
 for (int p=0;p<3;p++){
    for (int j=0;j<3;j++){
        cout<<"enter the element on "<<p+1<<" ; "<<p+1<<" position ";
        cin>>m[p][j];
         }
 }
 cout<<"entered matrix is : "<<endl;
for ( int p=0;p<3;p++ ){
    for ( int j=0;j<3;j++){
        cout<<m[p][j]<<" ";
}  cout<<endl;
    }

    for(int p =0; p<3; p++){
        l_f_d_s+= m[p][p];}
        cout<<" THE SUM OF LEFT DIAGONAL IS: "<<l_f_d_s<<endl;

        for( int p = 0; p<3; p++){
            r_d_s+= m[p][2-p];

        }
        cout<<"THE SUM OF RIGHT DIAGONAL SUM IS: "<<r_d_s<<endl;
        return 0;
    }
```

```
enter the element on 1 ; 1 position 2
enter the element on 1 ; 1 position 5
enter the element on 1 ; 1 position 9
enter the element on 2 ; 2 position 7
enter the element on 2 ; 2 position 4
enter the element on 2 ; 2 position 5
enter the element on 3 ; 3 position 6
enter the element on 3 ; 3 position 3
enter the element on 3 ; 3 position 2
entered matrix is :
2 5 9
7 4 5
6 3 2
 THE SUM OF LEFT DIAGONAL IS: 8
THE SUM OF RIGHT DIAGONAL SUM IS: 19

--------------------------------
Process exited after 10.7 seconds with return
 value 0
Press any key to continue . . .
```

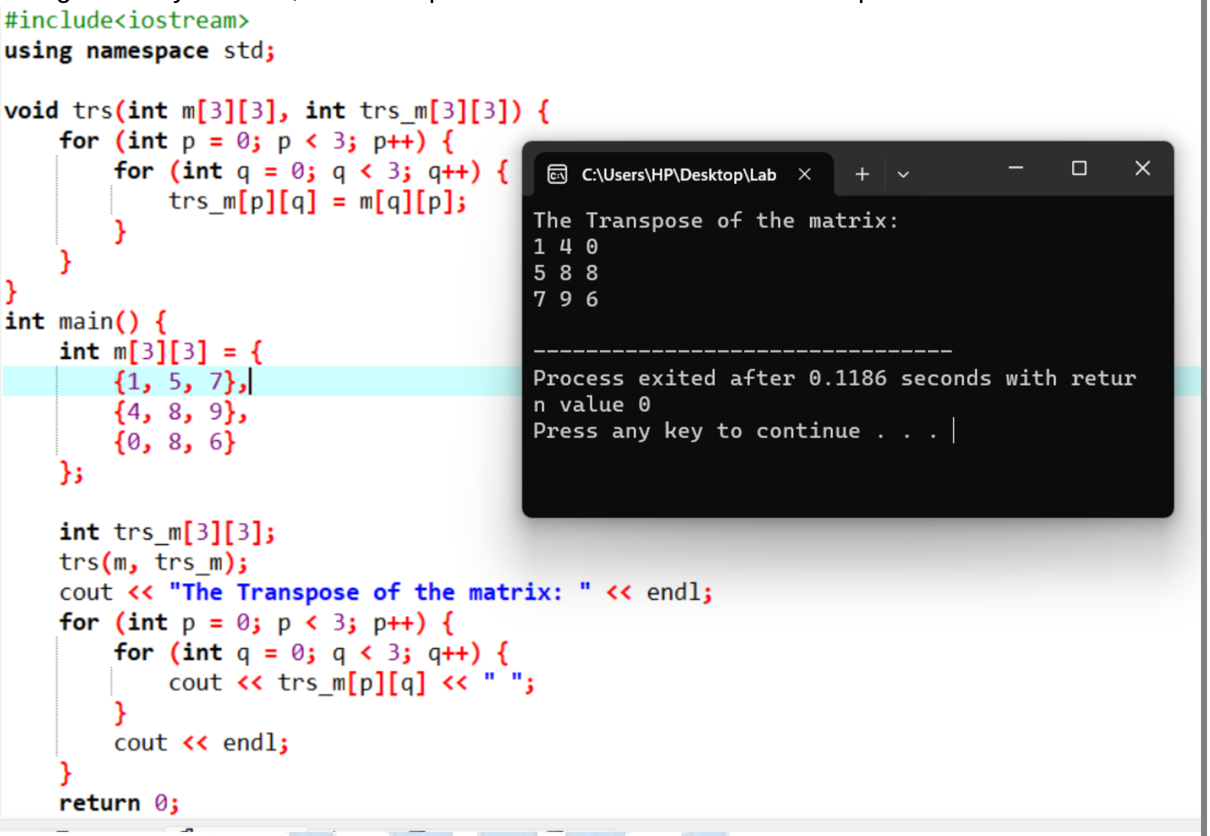
2. Write a function to add two 2D arrays of size 3x3.



```
#include<iostream>
using namespace std;
void add_arr(int arr1[3][3], int arr2[3][3], int res[3][3]) {
    for (int p = 0; p < 3; p++) {
        for (int q = 0; q < 3; q++) {
            res[p][q] = arr1[p][q] + arr2[p][q];
        }
    }
}

int main() {
    int arr1[3][3] = {
        {11, 222, 443},
        {64, 565, 446},
        {76, 78, 89} // Remove the semicolon here
    };

    int arr2[3][3] = {
        {10, 12, 13},
        {46, 55, 67},
        {71, 82, 98}
    };

    int res[3][3];
    add_arr(arr1, arr2, res);

    cout << "The Resultant array for this code is:" << endl;
    for (int p = 0; p < 3; p++) {
        for (int q = 0; q < 3; q++) {
            cout << res[p][q] << " ";
        }
        cout << endl;
    }

    return 0;
}
```

```
The Resultant array for this code is:
21 234 456
110 620 513
147 160 187

--------------------------------
Process exited after 0.1397 seconds with return value 0
Press any key to continue . . .
```

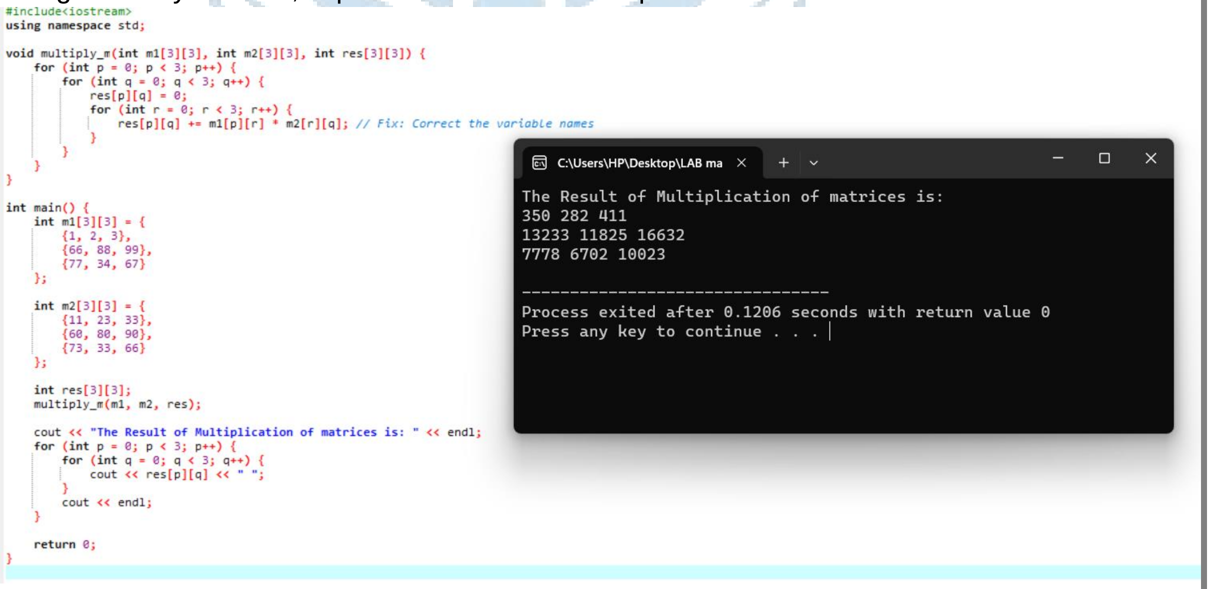
3. Using 2D arrays in C++, take transpose of a 3x3 matrix. Make a transpose function.



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

void trs(int m[3][3], int trs_m[3][3]) {
    for (int p = 0; p < 3; p++) {
        for (int q = 0; q < 3; q++) {
            trs_m[p][q] = m[q][p];
        }
    }
}
int main() {
    int m[3][3] = {
        {1, 5, 7},
        {4, 8, 9},
        {0, 8, 6}
    };

    int trs_m[3][3];
    trs(m, trs_m);
    cout << "The Transpose of the matrix: " << endl;
    for (int p = 0; p < 3; p++) {
        for (int q = 0; q < 3; q++) {
            cout << trs_m[p][q] << " ";
        }
        cout << endl;
    }
    return 0;
```

```
The Transpose of the matrix:
1 4 0
5 8 8
7 9 6

--------------------------------
Process exited after 0.1186 seconds with retur
n value 0
Press any key to continue . . .
```

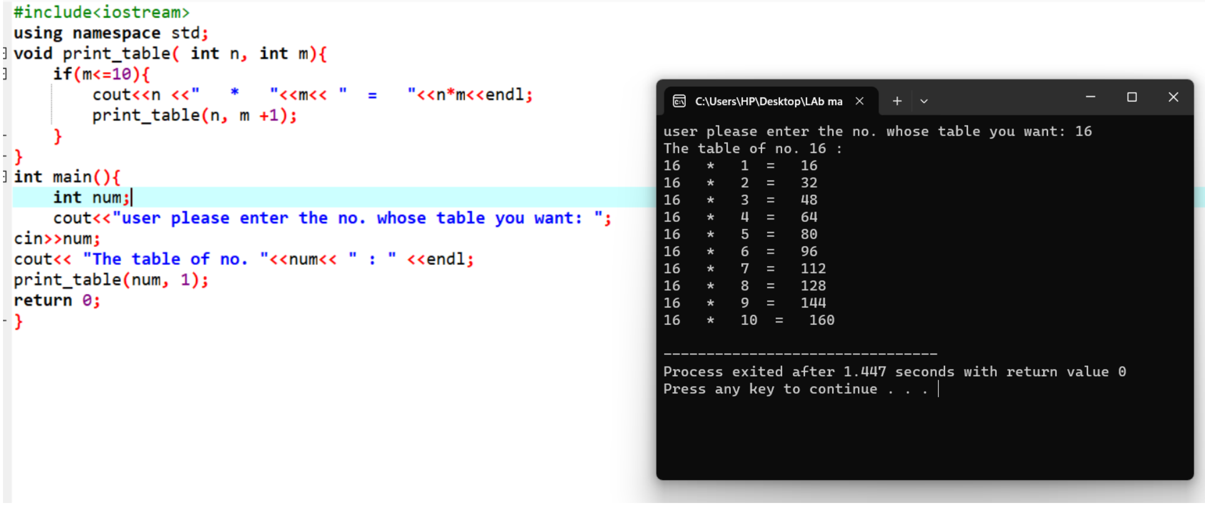
4. Using 2D arrays in C++, implement 3x3 matrix multiplication. Make a function.



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

void multiply_m(int m1[3][3], int m2[3][3], int res[3][3]) {
    for (int p = 0; p < 3; p++) {
        for (int q = 0; q < 3; q++) {
            res[p][q] = 0;
            for (int r = 0; r < 3; r++) {
                res[p][q] += m1[p][r] * m2[r][q]; // Fix: Correct the variable names
            }
        }
    }
}

int main() {
    int m1[3][3] = {
        {1, 2, 3},
        {66, 88, 99},
        {77, 34, 67}
    };

    int m2[3][3] = {
        {11, 23, 33},
        {60, 80, 90},
        {73, 33, 66}
    };

    int res[3][3];
    multiply_m(m1, m2, res);

    cout << "The Result of Multiplication of matrices is: " << endl;
    for (int p = 0; p < 3; p++) {
        for (int q = 0; q < 3; q++) {
            cout << res[p][q] << " ";
        }
        cout << endl;
    }

    return 0;
}
```

```
The Result of Multiplication of matrices is:
350 282 411
13233 11825 16632
7778 6702 10023

--------------------------------
Process exited after 0.1206 seconds with return value 0
Press any key to continue . . .
```

5. Print the multiplication table of 15 using recursion.



```cpp
#include<iostream>
using namespace std;
void print_table( int n, int m){
    if(m<=10){
        cout<<n <<"   *   "<<m<< "  =   "<<n*m<<endl;
        print_table(n, m +1);
    }
}
int main(){
    int num;
    cout<<"user please enter the no. whose table you want: ";
cin>>num;
cout<< "The table of no. "<<num<< " : " <<endl;
print_table(num, 1);
return 0;
}
```

```
user please enter the no. whose table you want: 16
The table of no. 16 :
16   *   1  =   16
16   *   2  =   32
16   *   3  =   48
16   *   4  =   64
16   *   5  =   80
16   *   6  =   96
16   *   7  =   112
16   *   8  =   128
16   *   9  =   144
16   *   10  =   160

--------------------------------
Process exited after 1.447 seconds with return value 0
Press any key to continue . . .
```

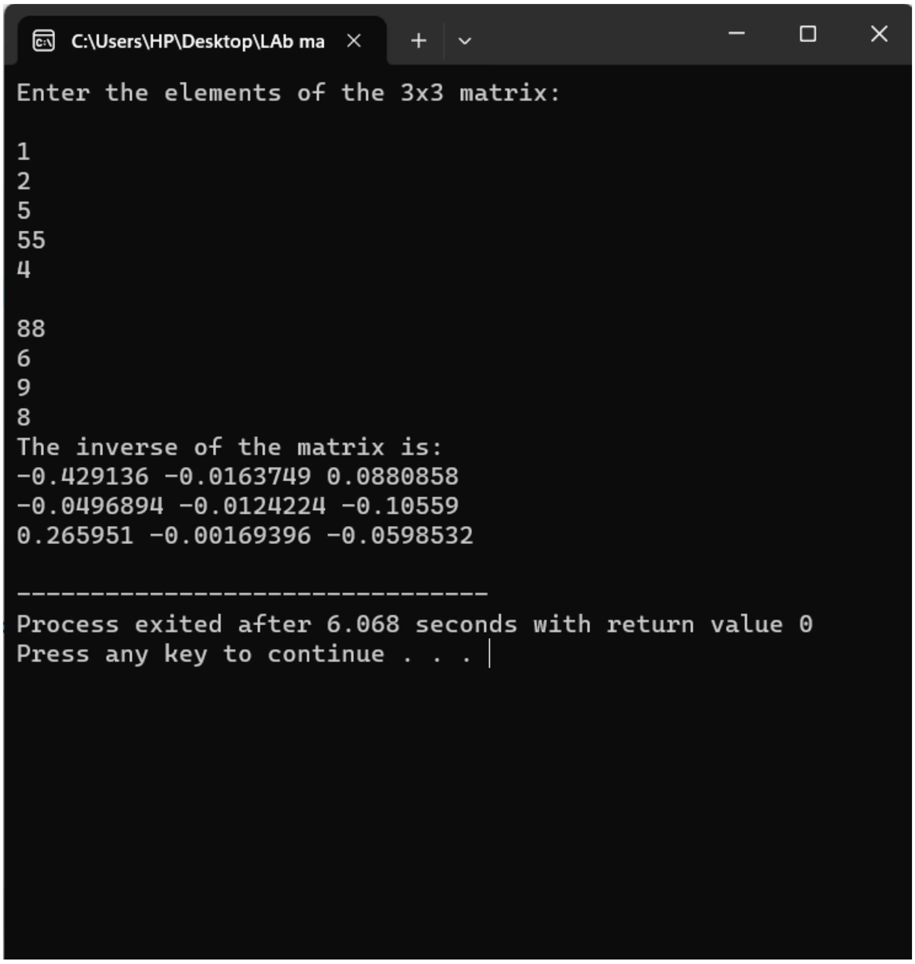
Home Task:

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

Ans:

```cpp
#include<iostream>
using namespace std;
double dt2x2(int a, int b, int c, int d) {
    return (a * d - b * c);
}
double dt3x3(int m[3][3]) {
    return (m[0][0] * dt2x2(m[1][1], m[1][2], m[2][1], m[2][2]) -
            m[0][1] * dt2x2(m[1][0], m[1][2], m[2][0], m[2][2]) +
            m[0][2] * dt2x2(m[1][0], m[1][1], m[2][0], m[2][1]));
}
void inverse(int m[3][3], double inv[3][3]) {
    double det = dt3x3(m);
    if (det == 0) {
        cout << "The matrix is singular. Inverse does not exist." << endl;
        return;
    }
    for (int p = 0; p < 3; p++) {
        for (int q = 0; q < 3; q++) {
            int cofactor_Sign = ((p + q) % 2 == 0) ? 1 : -1;
            int minorM[2][2] = {
                {m[(p + 1) % 3][(q + 1) % 3], m[(p + 1) % 3][(q + 2) % 3]},
                {m[(p + 2) % 3][(q + 1) % 3], m[(p + 2) % 3][(q + 2) % 3]}
            };
            double minorDet = dt2x2(minorM[0][0], minorM[0][1], minorM[1][0], minorM[1][1]);

            inv[q][p] = (cofactor_Sign * minorDet) / det;
        }
    }
    cout << "The inverse of the matrix is:" << endl;
    for (int p = 0; p < 3; p++) {
        for (int q = 0; q < 3; q++) {
            cout << inv[p][q] << " ";
        }
        cout << endl;
    }
}
int main() {
    int mat[3][3];
    cout << "Enter the elements of the 3x3 matrix:" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cin >> mat[i][j];
        }
    }
    double inv[3][3];
    inverse(mat, inv);
    return 0;
}
```

Output :



```
Enter the elements of the 3x3 matrix:

1
2
5
55
4

88
6
9
8
The inverse of the matrix is:
-0.429136 -0.0163749 0.0880858
-0.0496894 -0.0124224 -0.10559
0.265951 -0.00169396 -0.0598532

--------------------------------
Process exited after 6.068 seconds with return value 0
Press any key to continue . . .
```