

# CS & Programming Lab

## Lab Manual 09

---

Name: Abdullah Jamil Tung

Class: ME-15-Sec-A

Roll #: 478459

## **Lab Task #1**

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

```
#include<iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    //Declaring variables
```

```
    int a[3][3] , sum, x=1;
```

```
    //Taking Inputs
```

```
    cout<<"Enter the Elements for Matrix A\n\n";
```

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

```
    {
```

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

```
        {
```

```
            cout<<"Enter Element #"<<x<<" : ";
```

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

```
            x++;
```

```
}
```

```
}
```

```
cout<<"\n\n";
```

```
//Sum of Left Diagonal
```

```
cout<<"Left Diagonal\n\n";
```

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

```
{
```

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

```
{
```

```
if (i==j)
```

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

```
else
```

```
cout<<" ";
```

```
}
```

```
cout<<endl;
```

```
}
```

```
cout<<endl;
```

```
sum = a[0][0] + a[1][1] + a[2][2];
```

```
cout<<"Left Diagonal Sum = "<<a[0][0]<<" + "<<a[1][1]<<" +  
"<<a[2][2]<<" = "<<sum<<"\n\n\n";
```

```

//Sum of Right Diagonal

cout<<"Right Diagonal\n\n";

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

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

{
if (i==0&&j==2||i==1&&j==1||i==2&&j==0)

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

else

cout<<" ";

}

cout<<endl;

}

cout<<endl;

sum = a[0][2] + a[1][1] + a[2][0];

cout<<"Right Diagonal Sum = "<<a[0][2]<<" + "<<a[1][1]<<" +
"<<a[2][0]<<" = "<<sum<<"\n\n";

}

```

```

1  /*1. Make 2D Array in C++ and print left diagonal and right diagonal sum of a 3x3 matrix.*/
2
3  #include<iostream>
4  using namespace std;
5
6  int main()
7  {
8      //Declaring variables
9      int a[3][3] , sum, x=1;
10     //Taking Inputs
11     cout<<"Enter the Elements for Matrix A\n\n";
12     for (int i=0; i<3; i++)
13     {
14         for (int j=0; j<3; j++)
15         {
16             cout<<"Enter Element #"<<x<<" : ";
17             cin>>a[i][j];
18             x++;
19         }
20     }
21     cout<<"\n\n";
22
23     //Sum of Left Diagonal
24     cout<<"Left Diagonal\n\n";
25     for (int i=0; i<3; i++)
26     {
27         for (int j=0; j<3; j++)
28         {
29             if (i==j)
30                 cout<<a[i][j]<<" ";
31             else
32                 cout<<" ";
33         }
34         cout<<endl;
35     }
36     cout<<endl;
37
38     sum = a[0][0] + a[1][1] + a[2][2];
39     cout<<"Left Diagonal Sum = "<<a[0][0]<<" + "<<a[1][1]<<" + "<<a[2][2]<<" = "<<sum<<"\n\n";
40
41     //Sum of Right Diagonal
42     cout<<"Right Diagonal\n\n";
43     for (int i=0; i<3; i++)
44     {
45         for (int j=0; j<3; j++)
46         {
47             if (i==0&&j==2||i==1&&j==1||i==2&&j==0)
48                 cout<<a[i][j]<<" ";
49             else
50                 cout<<" ";
51         }
52         cout<<endl;
53     }
54     cout<<endl;
55     sum = a[0][2] + a[1][1] + a[2][0];
56     cout<<"Right Diagonal Sum = "<<a[0][2]<<" + "<<a[1][1]<<" + "<<a[2][0]<<" = "<<sum<<"\n\n";
57 }

```



C:\Users\SA\Downloads\ME-15-Sec-A\15

Enter the Elements for Matrix A

Enter Element #1 : 1  
Enter Element #2 : 2  
Enter Element #3 : 3  
Enter Element #4 : 4  
Enter Element #5 : 5  
Enter Element #6 : 6  
Enter Element #7 : 7  
Enter Element #8 : 8  
Enter Element #9 : 9

Left Diagonal

1  
  5  
   9

Left Diagonal Sum = 1 + 5 + 9 = 15

Right Diagonal

   3  
  5  
7

Right Diagonal Sum = 3 + 5 + 7 = 15

## **Lab Task #2**

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

```
#include<iostream>
```

```
using namespace std;
```

```
//Defining function
```

```
int sum(int a[3][3], int b[3][3], int c[3][3])
```

```
{
```

```
    //Computing Result
```

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

```
    {
```

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

```
        {
```

```
            c[i][j] = a[i][j] + b[i][j];
```

```
        }}
```

```
//Displaying Output
```

```
cout<<"Sum of 2D Matrices =\t";
```

```

        for (int i=0; i<3; i++)
        {
            for (int j=0; j<3; j++)
            {
                cout<<c[i][j]<<" ";
            }
            cout<<endl<<"\t\t";
        }
    }

```

//Main function

```

int main()
{
    //Declaring variables
    int a[3][3] = { {1,2,3} , {4,5,6} , {7,8,9} };
    int b[3][3] = { {1,2,3} , {4,5,6} , {7,8,9} };
    int c[3][3];

    //Calling function
    sum( a, b, c);
}

```



```

1  /*2. Write a function to add two 2D arrays of size 3x3.*/
2
3  #include<iostream>
4  using namespace std;
5
6  //Defining function
7  int sum(int a[3][3], int b[3][3], int c[3][3])
8  {
9      //Computing Result
10     for (int i=0; i<3; i++)
11     {
12         for (int j=0; j<3; j++)
13         {
14             c[i][j] = a[i][j] + b[i][j];
15         }
16     }
17     //Displaying Output
18     cout<<"Sum of 2D Matrices =\t";
19     for (int i=0; i<3; i++)
20     {
21         for (int j=0; j<3; j++)
22         {
23             cout<<c[i][j]<<" ";
24         }
25         cout<<endl<<"\t\t\t";
26     }
27
28     //Main function
29     int main()
30     {
31         //Declaring variables
32         int a[3][3] = {{1,2,3} , {4,5,6} , {7,8,9}};
33         int b[3][3] = {{1,2,3} , {4,5,6} , {7,8,9}};
34         int c[3][3];
35         //Calling function
36         sum( a, b, c);
37     }

```

```

C:\Users\SA\Downloads\ME-15-Sec-A
Sum of 2D Matrices =    2  4  6
                        8 10 12
                        14 16 18

```

## **Lab Task #3**

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

```
#include<iostream>
```

```
using namespace std;
```

```
//Defining function
```

```
int transpose (int a[3][3], int b[3][3])
```

```
{
```

```
    //Computing result
```

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

```
    {
```

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

```
        {
```

```
            b[j][i]=a[i][j];
```

```
        }
```

```
    }
```

```
}
```

```

//Main function

int main()
{
    //Declaring variables
    int a[3][3] = { {1,2,3},{4,5,6},{7,8,9}}, Transpose[3][3];
    //Calling function
    transpose (a,Transpose);
    //Displaying output
    cout<<"Transpose of a Matrix =\t";
    for (int i=0; i<3; i++)
    {
        for (int j=0; j<3; j++)
        {
            cout<<Transpose[i][j]<<" ";
        }
        cout<<endl<<"\t\t\t";
    }
}

```



## **Lab Task #4**

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

```
#include<iostream>
```

```
using namespace std;
```

```
//Defining function
```

```
int Multiply(int a[3][3], int b[3][3], int Multi[3][3])
```

```
{
```

```
    //Computing result
```

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

```
    {
```

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

```
        {
```

```
            Multi[i][j]=0;
```

```
            for (int k=0; k<3; k++)
```

```
            {
```

```
                Multi[i][j]+=a[i][k]*b[k][j];
```

```
            }
```

```

    }

    }

}

//Main function

int main()

{
    //Declaring variables

    int a[3][3]={ { 1,2,3},{4,5,6},{7,8,9} };
    int b[3][3]={ { 1,2,3},{4,5,6},{7,8,9} };
    int c[3][3];

    //Calling function

    Multiply(a,b,c);

    //Displaying output

    cout<<"Multiplication of Two 3x3 Matrices =\t";

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

    {

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

        {

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

        }

        cout<<endl<<"\t\t\t\t\t";
    }
}

```

```
}  
  
}
```

```
1  /*4. Using 2D arrays in C++, implement 3x3 matrix multiplication. Make a function.*/  
2  
3  #include<iostream>  
4  using namespace std;  
5  
6  //Defining function  
7  int Multiply(int a[3][3], int b[3][3], int Multi[3][3])  
8  {  
9      //Computing result  
10     for (int i=0; i<3; i++)  
11     {  
12         for (int j=0; j<3; j++)  
13         {  
14             Multi[i][j]=0;  
15             for (int k=0; k<3; k++)  
16             {  
17                 Multi[i][j]+=a[i][k]*b[k][j];  
18             }  
19         }  
20     }  
21 }  
22  
23 //Main function  
24 int main()  
25 {  
26     //Declaring variables  
27     int a[3][3]={{1,2,3},{4,5,6},{7,8,9}};  
28     int b[3][3]={{1,2,3},{4,5,6},{7,8,9}};  
29     int c[3][3];  
30     //Calling function  
31     Multiply(a,b,c);  
32     //Displaying output  
33     cout<<"Multiplication of Two 3x3 Matrices =\t";  
34     for (int i=0; i<3; i++)  
35     {  
36         for (int j=0; j<3; j++)  
37         {  
38             cout<<c[i][j]<<" ";  
39         }  
40         cout<<endl<<"\t\t\t\t\t";  
41     }  
42 }  
43
```

```
C:\Users\SA\Downloads\ME-15-Sec-A\1st Semester\Courses  
Multiplication of Two 3x3 Matrices =    30 36 42  
                                         66 81 96  
                                         102 126 150
```

## **Lab Task #5**

/\*5. Print the multiplication table of 15 using recursion.\*/

```
#include<iostream>
```

```
using namespace std;
```

```
//Making a recursive function
```

```
int Table_Of_15(int start, int end)
```

```
{
```

```
    //Checking for invalid operations
```

```
    if (start<=0)
```

```
        return 0;
```

```
    //Computing result
```

```
    else if (start<=end)
```

```
        {cout<<"15 x "<<start<<" = "<<15*start<<endl;
```

```
    //Recalling function
```

```
    return Table_Of_15(start+1, end);}
```

```
}
```



//Main Function

int main()

{

    //Declaring variables

    int x=1, y=10;

    //Displaying output

    cout<<"Table of 15\n\n";

    Table\_Of\_15(x,y);

}

```
1  /*5. Print the multiplication table of 15 using recursion.*/
2
3  #include<iostream>
4  using namespace std;
5
6  //Making a recursive function
7  int Table_Of_15(int start, int end)
8  {
9      //Checking for invalid operations
10     if (start<=0)
11         return 0;
12
13     //Computing result
14     else if (start<=end)
15     {cout<<"15 x "<<start<<" = "<<15*start<<endl;
16       //Recalling function
17       return Table_Of_15(start+1, end);}
18 }
19
20 //Main Function
21 int main()
22 {
23     //Declaring variables
24     int x=1, y=10;
25     //Displaying output
26     cout<<"Table of 15\n\n";
27     Table_Of_15(x,y);
28 }
```



C:\Users\SA\Downloads\ME-15-Sec-

## Table of 15

$$15 \times 1 = 15$$

$$15 \times 2 = 30$$

$$15 \times 3 = 45$$

$$15 \times 4 = 60$$

$$15 \times 5 = 75$$

$$15 \times 6 = 90$$

$$15 \times 7 = 105$$

$$15 \times 8 = 120$$

$$15 \times 9 = 135$$

$$15 \times 10 = 150$$

---

## **Home Task #1**

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

```
#include<iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    //Declaring variables
```

```
    float m[3][3], determ, det_1, det_2, det_3, cofac[3][3], adj[3][3],  
    inv[3][3], x=1;
```

```
    //Taking inputs
```

```
    cout<<"Enter the Elements for the 3x3 Matrix A\n"<<endl;
```

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

```
    {
```

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

```
        {
```

```
            cout<<"Enter Element #"<<x<<" = ";
```

```

    cin>>m[i][j];

    x++;

}

}

cout<<"\n\n";

//Calculating Determinant
det_1 = m[0][0] * (m[1][1]*m[2][2] - m[2][1]*m[1][2]);
det_2 = -m[0][1] * (m[1][0]*m[2][2] - m[2][0]*m[1][2]);
det_3 = m[0][2] * (m[1][0]*m[2][1] - m[2][0]*m[1][1]);
determ = det_1 + det_2 + det_3;
if (determ == 0)
    cout<<"Matrix cannot be inverted."<<endl;

else
{
    //Calculating Cofactor matrix
    cofac[0][0] = m[1][1]*m[2][2] - m[2][1]*m[1][2];
    cofac[0][1] = -(m[1][0]*m[2][2] - m[2][0]*m[1][2]);
    cofac[0][2] = m[1][0]*m[2][1] - m[2][0]*m[1][1];
    cofac[1][0] = -(m[0][1]*m[2][2] - m[2][1]*m[0][2]);
    cofac[1][1] = m[0][0]*m[2][2] - m[2][0]*m[0][2];

```

```
cofac[1][2] = -(m[0][0]*m[2][1] - m[2][0]*m[0][1]);
```

```
cofac[2][0] = m[0][1]*m[1][2] - m[1][1]*m[0][2];
```

```
cofac[2][1] = -(m[0][0]*m[1][2] - m[1][0]*m[0][2]);
```

```
cofac[2][2] = m[0][0]*m[1][1] - m[1][0]*m[0][1];
```

```
//Calculating Adjoint
```

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

```
{
```

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

```
{
```

```
    adj[j][i]=cofac[i][j];
```

```
}
```

```
}
```

```
//Displaying Matrix A
```

```
cout<<"Matrix A\n\n";
```

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

```
{
```

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

```
{
```

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

```

    }
    cout<<endl;
}
cout<<"\n\n";

//Displaying Inverse of Matrix A
cout<<"Inverse of Matrix A\n\n";
for (int i=0; i<3; i++)
{
    for (int j=0; j<3; j++)
    {
        //Computing Inverse of Matrix A
        inv[i][j] = adj[i][j]/determ;
        if (inv[i][j]==-0)
            inv[i][j]=0;
        cout<<inv[i][j]<<" ";
    }
    cout<<endl;
}
}
}
}

```

```

1  /*1. Write a C++ program to take inverse of a 3x3 matrix using its determinant and adjoint.*/
2
3  #include<iostream>
4  using namespace std;
5
6  int main()
7  {
8      //Declaring variables
9      float m[3][3], determ, det_1, det_2, det_3, cofac[3][3], adj[3][3], inv[3][3], x=1;
10
11     //Taking inputs
12     cout<<"Enter the Elements for the 3x3 Matrix A\n"<<endl;
13     for (int i=0; i<3; i++)
14     {
15         for (int j=0; j<3; j++)
16         {
17             cout<<"Enter Element #"<<x<<" = ";
18             cin>>m[i][j];
19             x++;
20         }
21     }
22     cout<<"\n\n";
23
24     //Calculating Determinant
25     det_1 = m[0][0] * (m[1][1]*m[2][2] - m[2][1]*m[1][2]);
26     det_2 = -m[0][1] * (m[1][0]*m[2][2] - m[2][0]*m[1][2]);
27     det_3 = m[0][2] * (m[1][0]*m[2][1] - m[2][0]*m[1][1]);
28     determ = det_1 + det_2 + det_3;
29     if (determ == 0)
30     cout<<"Matrix cannot be inverted."<<endl;
31

```

```

32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
else
{
    //Calculating Cofactor matrix
    cofac[0][0] = m[1][1]*m[2][2] - m[2][1]*m[1][2];
    cofac[0][1] = -(m[1][0]*m[2][2] - m[2][0]*m[1][2]);
    cofac[0][2] = m[1][0]*m[2][1] - m[2][0]*m[1][1];
    cofac[1][0] = -(m[0][1]*m[2][2] - m[2][1]*m[0][2]);
    cofac[1][1] = m[0][0]*m[2][2] - m[2][0]*m[0][2];
    cofac[1][2] = -(m[0][0]*m[2][1] - m[2][0]*m[0][1]);
    cofac[2][0] = m[0][1]*m[1][2] - m[1][1]*m[0][2];
    cofac[2][1] = -(m[0][0]*m[1][2] - m[1][0]*m[0][2]);
    cofac[2][2] = m[0][0]*m[1][1] - m[1][0]*m[0][1];

    //Calculating Adjoint
    for (int i=0; i<3; i++)
    {
        for (int j=0; j<3; j++)
        {
            adj[j][i]=cofac[i][j];
        }
    }

    //Displaying Matrix A
    cout<<"Matrix A\n\n";
    for (int i=0; i<3; i++)
    {
        for (int j=0; j<3; j++)
        {
            cout<<m[i][j]<<" ";
        }
        cout<<endl;
    }
    cout<<"\n\n";
}

```

```

65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
//Displaying Inverse of Matrix A
cout<<"Inverse of Matrix A\n\n";
for (int i=0; i<3; i++)
{
    for (int j=0; j<3; j++)
    {
        //Computing Inverse of Matrix A
        inv[i][j] = adj[i][j]/determ;
        if (inv[i][j]==-0)
            inv[i][j]=0;
        cout<<inv[i][j]<<" ";
    }
    cout<<endl;
}
}
}

```





C:\Users\SA\Downloads\ME-15-Sec-

```
Enter Element #2 = -1
Enter Element #3 = 0
Enter Element #4 = 0
Enter Element #5 = 1
Enter Element #6 = 2
Enter Element #7 = 1
Enter Element #8 = 1
Enter Element #9 = 0
```

Matrix A

```
2  -1  0
0   1  2
1   1  0
```

Inverse of Matrix A

```
0.333333  0  0.333333
-0.333333  0  0.666667
0.166667  0.5  -0.333333
```

-----