LAB MANUAL 09

LAB TASKS & HOME TASK

NAME: Muhammad Amaan Raza

CLASS: ME – 15 C

CMS ID: 465416

LAB TASKS:

Q 01 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 arr[3][3];

   int a=0, b=0;



   cout << "ENTER ELEMENTS OF THE MATRIX:" << endl;

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

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

         cin >> arr[i][j];

      }

   }

   cout << "ENTERED MATRIX:" << 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++) {
        a += arr[i][i];
        b += arr[i][2-i];
    }
            cout << "LEFT DIAGONAL SUM: " <<a<<endl;
    cout << "RIGHT DIAGONAL SUM: " <<b<<endl;


    return 0;
}
```

RESULT:

```
ENTER ELEMENTS OF THE MATRIX:
12
122
44
2
124
12
5
654
24
ENTERED MATRIX:
12 122 44
2 124 12
5 654 24
LEFT DIAGONAL SUM: 160
RIGHT DIAGONAL SUM: 173

--------------------------------
Process exited after 16.04 seconds with return value 0
Press any key to continue . . .
```

Q 02 Write a function to add two 2D arrays of size 3x3.

CODE:

```
#include <iostream>

using namespace std;



void add(int mat1[3][3], int mat2[3][3], int resultmat[3][3]) {

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

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

         resultmat[i][j] = mat1[i][j] + mat2[i][j];

      }
```

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


int main() {
    int mat1[3][3];
    int mat2[3][3];
    cout << "Enter elements of first matrix:" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cin >> mat1[i][j];
        }
```

```
    }

    cout << "Enter elements of second matrix:" << endl;

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

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

            cin >> mat2[i][j];

        }

    }



    int resultmat[3][3]; // To store the result

    add(mat1, mat2, resultmat);

    cout << "Result of matrix addition:" << endl;

    displaymat(resultmat);

    return 0;

}
```

Result:

```
Enter elements of first matrix:
234
2324
54
235
23
4323
43
4
42
Enter elements of second matrix:
4
24
5
76
89
65
13
56
344
Result of matrix addition:
238 2348 59
311 112 4388
56 60 386
```

Q 03 Using 2D arrays in C++, take transpose of a 3x3 matrix. Make a transpose function.

CODE:

```
#include<iostream>

using namespace std;

void transpose(int mat[3][3]) {

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

      for (int j=i+1;j<3;j++) {

         int temp=mat[i][j];

         mat[i][j]=mat[j][i];
```

```
            mat[j][i]=temp;
        }
    }
}
void display_matrix(int mat[3][3]) {
    for (int i=0; i<3;i++) {
        for (int j=0;j<3;j++) {
        cout<< mat[i][j] << " ";
        }
        cout<<endl;
    }
}
int main() {
    int mat[3][3];
    cout << "Enter elements of the matrix:"<<endl;
    for (int i=0; i<3;i++) {
        for (int j=0; j<3;j++) {
            cin >> mat[i][j];
        }
```

```
    }

    cout << "Original Matrix: "<<endl;

    display_matrix(mat);

    transpose(mat);

    cout << "Transposed Matrix: "<<endl;

    display_matrix(mat);

    return 0;

}
```

RESULT:

```
Enter elements of the matrix:
12
244
4
23
23
66
98
53
864
Original Matrix:
12 244 4
23 23 66
98 53 864
Transposed Matrix:
12 23 98
244 23 53
4 66 864
```

Q 04 Using 2D arrays in C++, implement 3x3 matrix multiplication. Make a function.

CODE:

```
#include <iostream>

using namespace std;



void multiplication(int m1[3][3], int m2[3][3], int result[3][3]) {

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

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

            result[i][j]=0;

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

                result[i][j] += m1[i][k] * m2[k][j];

    }

    }

    }

            }

void display_matrix(int mat[3][3]) {

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

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

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

        }

        cout<<endl;
```

```
    }

            }



int main() {

    int m1[3][3];

    int m2[3][3];



    cout << "Enter elements of the first matrix:" << endl;

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

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

            cin >> m1[i][j];

        }

                        }

                        cout << "Enter elements of the second matrix:" << endl;

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

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

            cin >> m2[i][j];

        }

                        }
```

```
    int result[3][3];

    multiplication(m1, m2, result);



    cout << "Result of matrix multiplication: " << endl;

    display_matrix(result);

    return 0;

}
```

RESULT:

```
Enter elements of the first matrix:
32
45
3
3
55
4
2
55
0
Enter elements of the second matrix:
34
3
3
55
23
55
23
2
6
Result of matrix multiplication:
3632 1137 2589
3219 1282 3058
3093 1271 3031
```

Q 05 Print the multiplication table of 15 using recursion.

Code:

```
#include <iostream>
```

```
using namespace std;



void Table(int num, int a) {

   if (a>10) {

      return;

   } else {

      int result = num*a;

      cout << num << " * " <<a<< " = " <<result<<endl;

      Table(num,a+1);

   }

   }

int main() {

   int num = 15;

   Table(num, 1);

         return 0;

}
```

RESULT:

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

--------------------------------
Process exited after 2.006 seconds with return value 0
Press any key to continue . . .
```

HOME TASK:

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

CODE:

```
#include <iostream>

using namespace std;

double Determinant(int mat[3][3]) {

   return mat[0][0]*(mat[1][1]*mat[2][2]-mat[1][2]*mat[2][1])-
mat[0][1]*(mat[1][0]*mat[2][2]-mat[1][2]*mat[2][0])+mat[0][2]*(mat[1][0]*mat[2][1]-
mat[1][1]*mat[2][0]);

}

void Adjoint(int mat[3][3], int adj[3][3]) {

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

```
        for (int j=0; j<3; j++) {
            adj[i][j] = (mat[(j+1)%3][(i + 1)%3]* mat[(j+2)%3][(i+2)%3]) -
                                (mat[(j+1)%3][(i+2)%3]*mat[(j+2)%3][(i+1)%3]);
        }
    }
            }
void Inverse(int mat[3][3], double inv[3][3]) {
    double det = Determinant(mat);
    if (det==0) {
        cout<<"Inverse does not exist (Matrix is singular)!"<<endl;
        return;
    }
    int 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]/det;
        }
    }
```

```
        }
int main() {
    int mat[3][3];
    cout << "Enter elements of the 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);
    if (Determinant(mat) != 0) {
    cout << "Inverse of the matrix:"<<endl;
    for (int i=0; i<3; i++) {
            for (int j=0; j<3;j++) {
                cout<<inv[i][j] << " ";
            }
            cout<<endl;
        }
```

```
    }

    return 0;

}
```

RESULT:

```
Enter elements of the matrix:
1
2
-1
-2
0
1
1
-1
0
Inverse of the matrix:
1 1 2
1 1 1
2 3 4
```