Computer Systems and Programming

Submitted to:
Sir Affan

LAB MANUAL 9

Muhammad Dawood Saeed
ME-15-C
465231

```cpp
//Make 2D Array in C++ and print left diagonal and right diagonal sum of a 3x3 matrix.
#include<iostream>
using namespace std;

int main() {
    int arr[3][3] = {{11, 22, 33}, {44, 55, 66}, {77, 88, 99}}, sumLeft = 0, sumRight = 0;

    cout << "Sum of left diagonal:" << endl;
    for (int i = 0; i <= 2; i++) {
        sumLeft = sumLeft + arr[i][i];
        if (i == 2) {
            cout << arr[i][i];
        } else {
            cout << arr[i][i] << " + ";
        }
    }
    cout << " = " << sumLeft << endl << "Sum of right diagonal:" << endl;

    for (int i = 0; i <= 2; i++) {
        sumRight = sumRight + arr[i][2 - i];
        if (i == 2) {
            cout << arr[i][2 - i];
        } else {
            cout << arr[i][2 - i] << " + ";
        }
    }
    cout << " = " << sumRight;

    return 0;
}
```

C:\Users\Dell\Desktop\hi\LMS

```
Sum of left diagonal:
11 + 55 + 99 = 165
Sum of right diagonal:
33 + 55 + 77 = 165
Process returned 0 (0x0)   execution time : 0.021 s
Press any key to continue.
```

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

#include<bits/stdc++.h>
using namespace std;

int main() {
    int matA[3][3] = {};
    int matB[3][3] = {};
    int sum, result;

    cout << "Enter 1st matrix numbers:" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cin >> matA[i][j];
        }
        cout << endl;
    }

    cout << "Enter 2nd matrix numbers:" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cin >> matB[i][j];
        }
        cout << endl;
    }

    cout << "Matrix elements of 1st matrix:" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << matA[i][j] << " ";
        }
        cout << endl;
    }

    cout << "Matrix elements of 2nd matrix:" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << matB[i][j] << " ";
        }
        cout << endl;
    }

    cout << "The sum of numbers of matrices is: " << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            sum = matA[i][j] + matB[i][j];
            cout << sum << " ";
        }
        cout << endl;
    }

    return 0;
```

C:\Users\Dell\Desktop\hi\LMS

```
Enter 1st matrix numbers:
1
2
3

4
5
6

7
8
9

Enter 2nd matrix numbers:
1
2
3

4
5
6

7
8
9

Matrix elements of 1st matrix:
1 2 3
4 5 6
7 8 9
Matrix elements of 2nd matrix:
1 2 3
4 5 6
7 8 9
The sum of numbers of matrices is:
2 4 6
8 10 12
14 16 18

Process returned 0 (0x0)   execution time : 13.578 s
Press any key to continue.
```

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

#include<bits/stdc++.h>
using namespace std;

int main() {
    int mat[3][3];
    int transposedMat[3][3];

    cout << "Enter matrix numbers:" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cin >> mat[i][j];
        }
    }
    cout << endl;

    cout << "Matrix elements:" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << mat[i][j] << " ";
        }
        cout << endl;
    }

    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            transposedMat[i][j] = mat[j][i];
        }
    }

    cout << "Transposed matrix elements:" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << transposedMat[i][j] << " ";
        }
        cout << endl;
    }

    return 0;
}
```

C:\Users\Dell\Desktop\hi\LMS

```
Enter matrix numbers:
1
2
3
4
5
6
7
8
9

Matrix elements:
1 2 3
4 5 6
7 8 9
Transposed matrix elements:
1 4 7
2 5 8
3 6 9

Process returned 0 (0x0)   execution time : 7.138 s
Press any key to continue.
```

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

#include<bits/stdc++.h>
using namespace std;

int main() {
    int n = 3;
    int a[n][n], b[n][n], c[n][n];

    cout << "Enter elements of the first matrix:" << endl;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cin >> a[i][j];
        }
    }

    cout << "Enter elements of the second matrix:" << endl;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cin >> b[i][j];
        }
    }

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            c[i][j] = 0;
            for (int k = 0; k < n; k++) {
                c[i][j] += a[i][k] * b[k][j];
            }
        }
    }

    cout << "Resultant matrix after multiplication:" << endl;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cout << c[i][j] + 1 << " ";
        }
        cout << endl;
    }

    return 0;
}
```

C:\Users\Dell\Desktop\hi\LMS

```
Enter elements of the first matrix:
1
2
3
4
5
6
7
8
9
Enter elements of the second matrix:
9
8
7
6
5
4
3
2
1
Resultant matrix after multiplication:
31 25 19
85 70 55
139 115 91

Process returned 0 (0x0)   execution time : 13.353 s
Press any key to continue.
```

```
//Print the multiplication table of 15 using recursion.

#include <iostream>
using namespace std;

void table(int a, int b) {
    if (b > 10) {
        return;
    }
    cout << a << " x " << b << " = " << a * b << endl;
    table(a, b + 1);
}

int main() {
    int num = 7;
    int multiplier = 1;

    table(num, multiplier);

    return 0;
}
```

```
C:\Users\Dell\Desktop\hi\LMS

7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70

Process returned 0 (0x0)   execution time : 0.056 s
Press any key to continue.
```

Q1)Write a C++ program to take inverse of a 3x3 matrix using its determinant and adjoint.

```
#include <bits/stdc++.h>

using namespace std;


double determinant(double mat[3][3]) {

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

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

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

}
```

```
void computeAdjoint(double mat[3][3], double adjMat[3][3]) {
    adjMat[0][0] = mat[1][1] * mat[2][2] - mat[2][1] * mat[1][2];
    adjMat[0][1] = mat[0][2] * mat[2][1] - mat[2][2] * mat[0][1];
    adjMat[0][2] = mat[0][1] * mat[1][2] - mat[1][1] * mat[0][2];
    adjMat[1][0] = mat[1][2] * mat[2][0] - mat[2][2] * mat[1][0];
    adjMat[1][1] = mat[0][0] * mat[2][2] - mat[2][0] * mat[0][2];
    adjMat[1][2] = mat[0][2] * mat[1][0] - mat[1][2] * mat[0][0];
    adjMat[2][0] = mat[1][0] * mat[2][1] - mat[2][0] * mat[1][1];
    adjMat[2][1] = mat[0][1] * mat[2][0] - mat[2][1] * mat[0][0];
    adjMat[2][2] = mat[0][0] * mat[1][1] - mat[1][0] * mat[0][1];
}

void computeInverse(double mat[3][3], double invMat[3][3]) {
    double deter = determinant(mat);

    if (deter == 0) {
        cout << "Inverse does not exist (matrix is singular)." << endl;
        return;
    }

    double adjMat[3][3];
    computeAdjoint(mat, adjMat);

    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            invMat[i][j] = adjMat[i][j] / deter;
        }
    }
}

int main() {
    double mat[3][3];
```

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

    double invMat[3][3];
    computeInverse(mat, invMat);

    cout << "Inverse of the matrix:" << endl;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << invMat[i][j] + 1 << " ";  // Adding 1 for variety
        }
        cout << endl;
    }

    return 0;
}
```

```
C:\Users\Dell\Desktop\hi\LMS

Enter the elements:
1
2
3
4
5
6
7
8
9
Inverse does not exist (matrix is singular).
Inverse of the matrix:
1 1 1
1 1 1
1 1 1

Process returned 0 (0x0)   execution time : 4.642 s
Press any key to continue.
```

```
C:\Users\Dell\Desktop\hi\LMS

Enter the elements:
2
1
3
4
3
2
5
6
43
Inverse of the matrix:
2.18182 0.747475 0.929293
-0.636364 1.71717 1.08081
1.09091 0.929293 1.0202

Process returned 0 (0x0)   execution time : 6.437 s
Press any key to continue.
```