COMPUTER SYSTEMS AND PROGRAMING

(lab task-9)

NAME: SAMEEN WASEEM

BATCH: ME-15

SECTION: A

CMS ID: 465064

SUBMITTED TO:

COURSE INSTRUCTOR: DR MUHAMMAD JAWAD KHAN

LAB INSTRUCTOR: SIR MUHAMMAD AFFAN

NUST

NATIONAL UNIVERSITY
OF SCIENCES & TECHNOLOGY

Code:

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

int main() {
    cout << "TASK-1" << endl;
    cout << "Sum of left and right diagonals of a 3x3 matrix" << endl;

    int matrix[3][3];
    int leftsum = 0;
    int rightsum = 0;

    cout << "Enter the elements of the 3x3 matrix:" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << "Enter element at position " << i + 1 << "," << j + 1 << ": ";
            cin >> matrix[i][j];
        }
    }

    for (int i = 0; i < 3; i++) {
        leftsum += matrix[i][i];
        rightsum += matrix[i][3 - 1 - i];
    }

    cout << "Sum of left diagonal: " << leftsum << endl;
    cout << "Sum of right diagonal: " << rightsum << endl;

    return 0;
}
```

Output:

```
TASK-1
Sum of left and right diagonals of a 3x3 matrix
Enter the elements of the 3x3 matrix:
Enter element at position 1,1: 1
Enter element at position 1,2: 1
Enter element at position 1,3: 1
Enter element at position 2,1: 2
Enter element at position 2,2: 2
Enter element at position 2,3: 2
Enter element at position 3,1: 3
Enter element at position 3,2: 3
Enter element at position 3,3: 3
Sum of left diagonal: 6
Sum of right diagonal: 6

--------------------------------
Process exited after 10.98 seconds with return value 0
Press any key to continue . . .
```

Code:

```cpp
#include<iostream>
using namespace std;
// Function to add two 3x3 matrices
void addMatrices(int mat1[3][3], int mat2[3][3], int sum[3][3]) {
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            sum[i][j] = mat1[i][j] + mat2[i][j];
        }
    }
}
int main() {
    cout << "TASK-2" << endl;
    cout << "Function to add two 2D arrays" << endl;
    int matrix1[3][3], matrix2[3][3], sum[3][3];
    // First matrix
    cout << "Enter the elements of the first 3x3 matrix:" << endl;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << "Enter element of first matrix: ";
            cin >> matrix1[i][j];
        }
    }
    // Second matrix
    cout << "Enter the elements of the second 3x3 matrix:" << endl;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << "Enter elements of second matrix: ";
            cin >> matrix2[i][j];
        }
    }
    addMatrices(matrix1, matrix2, sum);
    cout << "Sum of matrices:" << endl;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << sum[i][j] << " ";
        }
        cout << endl;
    }
 return 0;
}
```

Output:

```
TASK-2
Function to add two 2D arrays
Enter the elements of the first 3x3 matrix:
Enter element of first matrix: 1
Enter element of first matrix: 1
Enter element of first matrix: 1
Enter element of first matrix: 1
Enter element of first matrix: 1
Enter element of first matrix: 1
Enter element of first matrix: 1
Enter element of first matrix: 1
Enter element of first matrix: 1
Enter the elements of the second 3x3 matrix:
Enter elements of second matrix: 2
Enter elements of second matrix: 2
Enter elements of second matrix: 2
Enter elements of second matrix: 2
Enter elements of second matrix: 2
Enter elements of second matrix: 2
Enter elements of second matrix: 2
Enter elements of second matrix: 2
Enter elements of second matrix: 2
Sum of matrices:
3 3 3
3 3 3
3 3 3
```

Code:

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

// Function to find the transpose of a 3x3 matrix
void transposeMatrix(int mat[3][3], int transposedMat[3][3]) {
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            transposedMat[i][j] = mat[j][i];
        }
    }
}

int main() {
    cout << "TASK-3: Transpose of a 3x3 Matrix" << endl;

    int matrix[3][3], transposedMatrix[3][3];

    // Input matrix elements
    cout << "Enter the elements of the 3x3 matrix:" << endl;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << "Enter element at position " << i + 1 << "," << j + 1 << ": ";
            cin >> matrix[i][j];
        }
    }

    // Call the transpose function
    transposeMatrix(matrix, transposedMatrix);

    // Display the original matrix
    cout << "Original Matrix:" << endl;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << matrix[i][j] << " ";
        }
        cout << endl;
    }

    // Display the transposed matrix
    cout << "Transposed Matrix:" << endl;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << transposedMatrix[i][j] << " ";
        }
        cout << endl;
    }

    return 0;
}
```

Output:

```
TASK-3: Transpose of a 3x3 Matrix
Enter the elements of the 3x3 matrix:
Enter element at position 1,1: 1
Enter element at position 1,2: 2
Enter element at position 1,3: 3
Enter element at position 2,1: 4
Enter element at position 2,2: 5
Enter element at position 2,3: 6
Enter element at position 3,1: 7
Enter element at position 3,2: 8
Enter element at position 3,3: 9
Original Matrix:
1 2 3
4 5 6
7 8 9
Transposed Matrix:
1 4 7
2 5 8
3 6 9

--------------------------------
Process exited after 22.27 seconds with return value 0
Press any key to continue . . .
```

Code:

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

// Function to multiply two 3x3 matrices
void multiplyMatrices(int mat1[3][3], int mat2[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] += mat1[i][k] * mat2[k][j];
            }
        }
    }
}
int main() {
    cout << "TASK-4: Matrix Multiplication for 3x3 Matrices" << endl;

    int matrix1[3][3], matrix2[3][3], resultMatrix[3][3];
    //first matrix
    cout << "Enter the elements of the first 3x3 matrix:" << endl;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << "Enter element at position " << i + 1 << "," << j + 1 << ": ";
            cin >> matrix1[i][j];
        }
    }
    //second matrix
    cout << "Enter the elements of the second 3x3 matrix:" << endl;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << "Enter element at position " << i + 1 << "," << j + 1 << ": ";
            cin >> matrix2[i][j];
        }
    }
    multiplyMatrices(matrix1, matrix2, resultMatrix);
    //after multiplication
    cout << "Resultant Matrix after Multiplication:" << endl;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << resultMatrix[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

Output:

```
TASK-4: Matrix Multiplication for 3x3 Matrices
Enter the elements of the first 3x3 matrix:
Enter element at position 1,1: 1
Enter element at position 1,2: 1
Enter element at position 1,3: 1
Enter element at position 2,1: 1
Enter element at position 2,2: 1
Enter element at position 2,3: 1
Enter element at position 3,1: 1
Enter element at position 3,2: 1
Enter element at position 3,3: 1
Enter the elements of the second 3x3 matrix:
Enter element at position 1,1: 2
Enter element at position 1,2: 3
Enter element at position 1,3: 4
Enter element at position 2,1: 2
Enter element at position 2,2: 3
Enter element at position 2,3: 4
Enter element at position 3,1: 5
Enter element at position 3,2: 6
Enter element at position 3,3: 7
Resultant Matrix after Multiplication:
9 12 15
9 12 15
9 12 15

--------------------------------
Process exited after 30.3 seconds with return value 0
Press any key to continue . . .
```

Code:

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

void printMultiplicationTable(int x, int multiplier) {
    if (multiplier > 10) {
        cout<<"the recursion should take place till 10"<<endl;
        return;
    }

    cout << x << " * " << multiplier << " = " << x * multiplier << endl;

    // Recursively call the function with the next multiplier
    printMultiplicationTable(x , multiplier + 1);
}

int main() {
    cout<<"TASK-5"<<endl;
    int x = 15;

    cout << "Multiplication table of " << x << ":" << endl;
    printMultiplicationTable(x, 1);
    return 0;
}
```

Output:

```
TASK-5
Multiplication table of 15:
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
the recursion should take place till 10

--------------------------------
Process exited after 0.5681 seconds with return value 0
Press any key to continue . . .
```

Code:

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

// Function to calculate the determinant of a 3x3 matrix
int determinantOfMatrix(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]);
}

//the cofactor of a matrix
void Cofactor(int mat[3][3], int temp[3][3], int p, int q, int n) {
    int i = 0, j = 0;

    for (int row = 0; row < n; row++) {
        for (int col = 0; col < n; col++) {
            if (row != p && col != q) {
                temp[i][j++] = mat[row][col];

                if (j == n - 1) {
                    j = 0;
                    i++;
                }
            }
        }
    }
}

```

```
29  // the adjoint of a 3x3 matrix
30  void adjointOfMatrix(int mat[3][3], int adj[3][3]) {
31      for (int i = 0; i < 3; i++) {
32          for (int j = 0; j < 3; j++) {
33              int sign = ((i + j) % 2 == 0) ? 1 : -1;
34              int temp[3][3];
35              Cofactor(mat, temp, i, j, 3);
36              adj[j][i] = sign * determinantOfMatrix(temp);
37          }
38      }
39  }
40
41  //the inverse of a 3x3 matrix
42  bool inverseOfMatrix(int mat[3][3], double inv[3][3]) {
43      int det = determinantOfMatrix(mat);
44
45      if (det == 0) {
46          cout << "The matrix is singular and doesn't have an inverse." << endl;
47          return false;
48      }
49
50      int adj[3][3];
51      adjointOfMatrix(mat, adj);
52
53      for (int i = 0; i < 3; i++) {
54          for (int j = 0; j < 3; j++) {
55              inv[i][j] = adj[i][j] / double(det);
56          }
57      }
58
58
59      return true;
60  }
```

Output:

```
Enter the elements of the 3x3 matrix:
Enter element at position 1,1: 1
Enter element at position 1,2: 2
Enter element at position 1,3: 3
Enter element at position 2,1: 1
Enter element at position 2,2: 2
Enter element at position 2,3: 3
Enter element at position 3,1: 1
Enter element at position 3,2: 2
Enter element at position 3,3: 3
The matrix is singular and doesn't have an inverse.

--------------------------------
Process exited after 15.3 seconds with return value 0
Press any key to continue . . .
```