Question –

WAP to create a class MATRIX and perform the following functions:

1. Input

2. Add

3. Subtract

4. Transpose

5 Multiply

6 Display

Code –

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

class MATRIX
{
private:
    int rows, cols;
    int **matrix;

public:
    MATRIX(int r, int c)
    {
        rows = r;
        cols = c;
        matrix = new int *[rows];
        for (int i = 0; i < rows; i++)
        {
            matrix[i] = new int[cols];
        }
    }

    void input()
    {
        cout << "Enter matrix elements (" << rows << "x" << cols << "):" << endl;
        for (int i = 0; i < rows; i++)
        {
            for (int j = 0; j < cols; j++)
```

```
            {
                cin >> matrix[i][j];
            }
        }
    }

    void display()
    {
        cout << "Matrix (" << rows << "x" << cols << "):" << endl;
        for (int i = 0; i < rows; i++)
        {
            for (int j = 0; j < cols; j++)
            {
                cout << matrix[i][j] << " ";
            }
            cout << endl;
        }
    }

    MATRIX add(MATRIX &m)
    {
        if (rows != m.rows || cols != m.cols)
        {
            cout << "Matrices cannot be added. Dimensions do not match." << endl;
            return MATRIX(0, 0);
        }
        MATRIX result(rows, cols);
        for (int i = 0; i < rows; i++)
        {
            for (int j = 0; j < cols; j++)
            {
                result.matrix[i][j] = matrix[i][j] + m.matrix[i][j];
            }
        }
        return result;
    }

    MATRIX subtract(MATRIX &m)
    {
        if (rows != m.rows || cols != m.cols)
        {
            cout << "Matrices cannot be subtracted. Dimensions do not match." <<
endl;
            return MATRIX(0, 0);
        }
```

```
        MATRIX result(rows, cols);
        for (int i = 0; i < rows; i++)
        {
            for (int j = 0; j < cols; j++)
            {
                result.matrix[i][j] = matrix[i][j] - m.matrix[i][j];
            }
        }
        return result;
    }

    MATRIX transpose()
    {
        MATRIX result(cols, rows);
        for (int i = 0; i < cols; i++)
        {
            for (int j = 0; j < rows; j++)
            {
                result.matrix[i][j] = matrix[j][i];
            }
        }
        return result;
    }

    MATRIX multiply(MATRIX &m)
    {
        if (cols != m.rows)
        {
            cout << "Matrices cannot be multiplied. Dimensions do not match." <<
endl;
            return MATRIX(0, 0);
        }
        MATRIX result(rows, m.cols);
        for (int i = 0; i < rows; i++)
        {
            for (int j = 0; j < m.cols; j++)
            {
                result.matrix[i][j] = 0;
                for (int k = 0; k < cols; k++)
                {
                    result.matrix[i][j] += matrix[i][k] * m.matrix[k][j];
                }
            }
        }
        return result;
```

```
    }
};

int main()
{
    int r1, c1, r2, c2;

    cout << "Enter dimensions of first matrix (rows columns): ";
    cin >> r1 >> c1;
    MATRIX mat1(r1, c1);
    mat1.input();

    cout << "Enter dimensions of second matrix (rows columns): ";
    cin >> r2 >> c2;
    MATRIX mat2(r2, c2);
    mat2.input();

    cout << "\nFirst Matrix:" << endl;
    mat1.display();

    cout << "\nSecond Matrix:" << endl;
    mat2.display();

    cout << "\nAddition of Matrices:" << endl;
    MATRIX addResult = mat1.add(mat2);
    addResult.display();

    cout << "\nSubtraction of Matrices:" << endl;
    MATRIX subResult = mat1.subtract(mat2);
    subResult.display();

    cout << "\nTranspose of First Matrix:" << endl;
    MATRIX transposeResult = mat1.transpose();
    transposeResult.display();

    cout << "\nMultiplication of Matrices:" << endl;
    MATRIX mulResult = mat1.multiply(mat2);
    mulResult.display();

    return 0;
}
```

Output –

```
Enter dimensions of first matrix (rows columns): 2 2
Enter matrix elements (2x2):
1
3
4
8
Enter dimensions of second matrix (rows columns): 2 2
Enter matrix elements (2x2):
4
5
6
7

First Matrix:
Matrix (2x2):
1 3
4 8

Second Matrix:
Matrix (2x2):
4 5
6 7

Addition of Matrices:
Matrix (2x2):
5 8
10 15

Subtraction of Matrices:
Matrix (2x2):
-3 -2
-2 1

Transpose of First Matrix:
Matrix (2x2):
1 4
3 8

Multiplication of Matrices:
Matrix (2x2):
22 26
64 76
```