**Practical 6**

**Name: Saurabh Asnare (SYCOA 276)**

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

#define N 4
int flag = 0;

// # Matrix Class
class Matrix
{
private:
    int rows, col;
    int arr[N][N];

public:
    void acceptMatrix(int);
    void displayMatrix(int);
    void Addition(Matrix, Matrix);
    void Subtraction(Matrix, Matrix);
    void Multiplication(Matrix, Matrix);
    Matrix()
    {
        cout << "Matrix class object created.." << endl;
        rows = 0;
        col = 0;
    }
    ~Matrix()
    {
        rows = 0;
        col = 0;
    }
};

// *** Function Defination ***

void Matrix ::acceptMatrix(int k)
{
    int i, j;
    flag = 2;
    cout << "\n# Matrix-" << k << " : " << endl;
    cout << "Enter rows: ";
    cin >> rows;
    cout << "Enter columns: ";
    cin >> col;
    try
    {
        if (rows == 0 || col == 0 || rows > N || col > N)
        {
            throw 1;
        }
        {
            cout << "\n* Enter matrix-" << k << " : " << endl;
            for (i = 0; i < rows; i++)
            {
                for (j = 0; j < col; j++)
                {
```

```
                cin >> arr[i][j];
            }
        }
    }
  }
  catch (int e)
  {
    cout << "\n* Size Overflow.." << endl;
    flag = 1;
  }
}

void Matrix ::displayMatrix(int k)
{
  int i, j;
  cout << "\n# Matrix-" << k << " is: " << endl;
  if (flag == 1 || flag == 0)
  {
    cout << "* Please Enter matrix first..." << endl;
  }
  else
  {
    for (i = 0; i < rows; i++)
    {
      for (j = 0; j < col; j++)
      {
        cout << arr[i][j] << " ";
      }
      cout << endl;
    }
  }
}
void Matrix ::Addition(Matrix m1, Matrix m2)
{
  // Resultant matrix is
  int i, j;
  if (flag == 1 || flag == 0)
  {
    cout << "\n* Enter Matrix first" << endl;
  }
  else
  {
    try
    {
      if (m1.rows != m2.rows | m1.col != m2.col)
      {
        throw 1;
      }
      {
        cout << "\n# Resultant matrix is: " << endl;
        for (i = 0; i < m1.rows; i++)
        {
          for (j = 0; j < m1.col; j++)
          {
            arr[i][j] = m1.arr[i][j] + m2.arr[i][j];
```

```
                cout << arr[i][j] << " ";
            }
            cout << endl;
        }
    }
}

catch (int e)
{
    cout << "\n* Matrix order should be same.." << endl;
}
}
}

void Matrix ::Subtraction(Matrix m1, Matrix m2)
{
  // Resultant matrix is
  int i, j;
  if (flag == 1 || flag == 0)
  {
    cout << "\n* Enter Matrix first" << endl;
  }
  else
  {
    try
    {
      if (m1.rows != m2.rows | m1.col != m2.col)
      {
        throw 1;
      }
      {
        cout << "\n# Resultant matrix is: " << endl;
        for (i = 0; i < m1.rows; i++)
        {
          for (j = 0; j < m1.col; j++)
          {
            arr[i][j] = m1.arr[i][j] - m2.arr[i][j];
            cout << arr[i][j] << " ";
          }
          cout << endl;
        }
      }
    }
    catch (int e)
    {
      cout << "\n* Order should be same.." << endl;
    }
  }
}

void Matrix ::Multiplication(Matrix m1, Matrix m2)
{
  int i, j, k, sum;
  if (flag == 1 || flag == 0)
  {
```

```
        cout << "\n* Enter Matrix first" << endl;
    }
    else
    {
        try
        {
            if (m1.col != m2.rows)
            {
                throw 1;
            }
            {
                for (i = 0; i < m1.rows; i++)
                {
                    for (j = 0; j < m2.col; j++)
                    {
                        sum = 0;
                        for (k = 0; k < m1.col; k++)
                        {
                            sum = sum + (m1.arr[i][k] * m2.arr[k][j]);
                        }
                        arr[i][j] = sum;
                    }
                }
                printf("\n# Resultant matrix is: \n");
                for (i = 0; i < m1.rows; i++)
                {
                    for (j = 0; j < m2.col; j++)
                    {
                        cout << arr[i][j] << " ";
                    }
                    cout << endl;
                }
            }
        }
        catch (int e)
        {
            cout << "\n* You can't multiply.." << endl;
        }
    }
}

// # Main Function
int main()
{
    cout << endl;
    Matrix m1, m2, m;
    cout << "\n # Matrix Arithmetic #\n";
    int ch;
    do
    {
        cout << "\n# Select from below choices: \n1.Accept Matrix \n2.Display Matrix \n3.Addition \n4.Subtraction \n5.Multiplication \n6.Exit \nEnter your choice:  ";
        cin >> ch;
        switch (ch)
        {
        case 1:
```

```
            m1.acceptMatrix(1);
            m2.acceptMatrix(2);
            break;

        case 2:
            m1.displayMatrix(1);
            m2.displayMatrix(2);
            break;

        case 3:
            m.Addition(m1, m2);
            break;

        case 4:
            m.Subtraction(m1, m2);
            break;

        case 5:
            m.Multiplication(m1, m2);
            break;

        case 6:
            cout << "\n* You are exit!\n";
            cout << endl;
            exit(0);
            break;

        default:
            cout << "\n* Invalid choice!" <<
endl;
            break;
        }
    } while (1);
    return 0;
}
```

## Output:

```
Matrix class object created..
Matrix class object created..
Matrix class object created..

 # Matrix Arithmetic #

# Select from below choices:
1.Accept Matrix
2.Display Matrix
3.Addition
4.Subtraction
5.Multiplication
6.Exit
Enter your choice:  1

# Matrix-1 :
Enter rows: 2
Enter columns: 2

* Enter matrix-1 :
1
2
3
4
```

# Matrix-2 :

Enter rows: 2

Enter columns: 2

* Enter matrix-2 :

5

6

7

8

# Select from below choices:

1.Accept Matrix

2.Display Matrix

3.Addition

4.Subtraction

5.Multiplication

6.Exit

Enter your choice:  3

# Resultant matrix is:

6 8

10 12

# Select from below choices:

1.Accept Matrix

2.Display Matrix

3.Addition

4.Subtraction

5.Multiplication

6.Exit

Enter your choice:  4

# Resultant matrix is:

-4 -4

-4 -4

# Select from below choices:

1.Accept Matrix

2.Display Matrix

3.Addition

4.Subtraction

5.Multiplication

6.Exit

Enter your choice:  5

# Resultant matrix is:

19 22

43 50