LAB-1

Q1. Write a C program for

1. **Pass the matrices as parameters**
2. **Addition\Substraction**
3. **Sum of rows & columns**
4. **Multiplication**
5. **Sum of principle\non principle diagonal elements**
6. **Transpose of matrix**
7. **Symmetric or not**

Aim : To execute all the above operations.

CODE:

```
#include <stdio.h>

#define MAX_SIZE 100


// Function to input a matrix
void inputMatrix(int matrix[MAX_SIZE][MAX_SIZE], int rows, int cols) {
printf("Enter the elements of the matrix:\n");     for (int i = 0; i < rows; i++)
{        for (int j = 0; j < cols; j++) {            scanf("%d", &matrix[i][j]);
      }
   }
}


// Function to print a matrix
void printMatrix(int matrix[MAX_SIZE][MAX_SIZE], int rows, int cols) {
printf("Matrix:\n");     for (int i = 0; i < rows; i++) {         for (int j = 0; j <
cols; j++) {            printf("%d ", matrix[i][j]);
      }
printf("\n");
   }
}


// Function to add two matrices
```

```
void addMatrices(int matrix1[MAX_SIZE][MAX_SIZE], int matrix2[MAX_SIZE][MAX_SIZE], int
rows, int cols) {     int result[MAX_SIZE][MAX_SIZE];
   for (int i = 0; i < rows; i++) {         for (int j =
0; j < cols; j++) {             result[i][j] =
matrix1[i][j] + matrix2[i][j];
       }
   }
   printf("Addition of matrices:\n");
printMatrix(result, rows, cols);
}

// Function to subtract two matrices
void subtractMatrices(int matrix1[MAX_SIZE][MAX_SIZE], int
matrix2[MAX_SIZE][MAX_SIZE], int rows, int cols) {     int
result[MAX_SIZE][MAX_SIZE];
   for (int i = 0; i < rows; i++) {         for (int j =
0; j < cols; j++) {             result[i][j] =
matrix1[i][j] - matrix2[i][j];
       }
   }
   printf("Subtraction of matrices:\n");
printMatrix(result, rows, cols);
}

// Function to multiply two matrices
void multiplyMatrices(int matrix1[MAX_SIZE][MAX_SIZE], int rows1, int cols1, int
matrix2[MAX_SIZE][MAX_SIZE], int rows2, int cols2) {
   if (cols1 != rows2) {         printf("Error: Matrices
cannot be multiplied.\n");         return;
   }

   int result[MAX_SIZE][MAX_SIZE];
```

```
    for (int i = 0; i < rows1; i++) {
        for (int j = 0; j < cols2; j++) {            result[i][j]
= 0;            for (int k = 0; k < cols1; k++) {
result[i][j] += matrix1[i][k] * matrix2[k][j];
            }
        }
    }
    printf("Multiplication of matrices:\n");
printMatrix(result, rows1, cols2);
}

// Function to calculate the sum of diagonal or non-diagonal elements in a matrix void
sumDiagonalNonDiagonal(int matrix[MAX_SIZE][MAX_SIZE], int rows, int cols, char choice)
{    int sum = 0;    if (choice == 'D'
|| choice == 'd') {        for (int i = 0;
i < rows; i++) {            sum +=
matrix[i][i];
        }
        printf("Sum of diagonal elements: %d\n", sum);
    } else if (choice == 'N' || choice == 'n') {
for (int i = 0; i < rows; i++) {            for
(int j = 0; j < cols; j++) {                if (i !=
j) {                    sum += matrix[i][j];
                }
            }
        }
        printf("Sum of non-diagonal elements: %d\n", sum);
    } else {        printf("Invalid choice. Please enter
D or N.\n");
    }
}
```

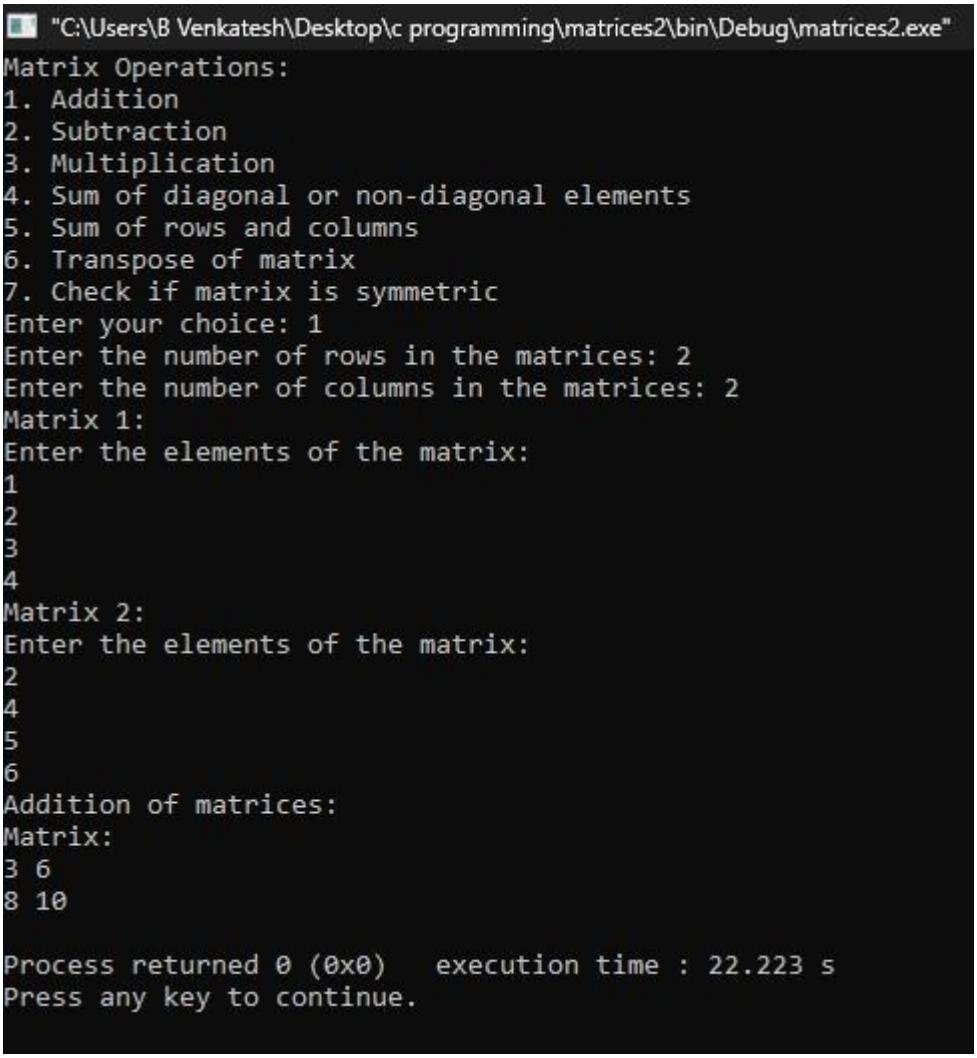
```
// Function to calculate the sum of rows and columns in a matrix void
sumRowsColumns(int matrix[MAX_SIZE][MAX_SIZE], int rows, int cols) {
int rowSum[MAX_SIZE] = {0};     int colSum[MAX_SIZE] = {0};

   for (int i = 0; i < rows; i++) {
for (int j = 0; j < cols; j++) {
rowSum[i] += matrix[i][j];
colSum[j] += matrix[i][j];
       }
   }

   printf("Sum of rows:\n");     for (int i = 0; i <
rows; i++) {         printf("Row %d: %d\n", i + 1,
rowSum[i]);
   }

   printf("Sum of columns:\n");     for (int j = 0; j
< cols; j++) {         printf("Column %d: %d\n", j +
1, colSum[j]);
   }
}

// Function to transpose a matrix
void transposeMatrix(int matrix[MAX_SIZE][MAX_SIZE], int rows, int cols) {
int transposed[MAX_SIZE][MAX_SIZE];
   for (int i = 0; i < rows; i++) {
for (int j = 0; j < cols; j++) {
transposed[j][i] = matrix[i][j];
       }
   }
```

```
    printf("Transposed matrix:\n");
printMatrix(transposed, cols, rows);
}

// Function to check if a matrix is symmetric
int isSymmetricMatrix(int matrix[MAX_SIZE][MAX_SIZE], int rows, int cols) {
    if (rows != cols) {
return 0;
    }

    for (int i = 0; i < rows; i++) {
for (int j = 0; j < cols; j++) {            if
(matrix[i][j] != matrix[j][i]) {
return 0;
            }
        }
    }

    return 1;
}

int main() {     int choice;     printf("Matrix Operations:\n");
printf("1.  Addition\n");             printf("2.  Subtraction\n");
printf("3. Multiplication\n");      printf("4. Sum of diagonal
or non-diagonal elements\n");     printf("5. Sum of rows and
columns\n");             printf("6.  Transpose  of  matrix\n");
printf("7. Check if matrix is symmetric\n");     printf("Enter
your choice: ");     scanf("%d", &choice);

    int rows, cols;     printf("Enter the number of rows in
the matrices: ");     scanf("%d", &rows);
```

```
   printf("Enter the number of columns in the matrices: ");
scanf("%d", &cols);

   int matrix1[MAX_SIZE][MAX_SIZE];
int matrix2[MAX_SIZE][MAX_SIZE];

   switch (choice) {
case 1:
        printf("Matrix 1:\n");
inputMatrix(matrix1, rows, cols);
printf("Matrix 2:\n");          inputMatrix(matrix2,
rows, cols);          addMatrices(matrix1, matrix2,
rows, cols);
        break;
case 2:
        printf("Matrix 1:\n");
inputMatrix(matrix1, rows, cols);
printf("Matrix 2:\n");          inputMatrix(matrix2,
rows, cols);          subtractMatrices(matrix1, matrix2,
rows, cols);
        break;
case 3:
        printf("Matrix 1:\n");          inputMatrix(matrix1, rows,
cols);          printf("Matrix 2:\n");          inputMatrix(matrix2,
cols, rows);          multiplyMatrices(matrix1, rows, cols, matrix2,
cols, rows);
        break;
case 4:
        printf("Matrix:\n");          inputMatrix(matrix1, rows, cols);
printf("Enter 'D' for diagonal elements or 'N' for non-diagonal elements: ");
char sumChoice;          scanf(" %c", &sumChoice);
```

```
            sumDiagonalNonDiagonal(matrix1, rows, cols, sumChoice);
            break;
case 5:
            printf("Matrix:\n");
inputMatrix(matrix1, rows, cols);
sumRowsColumns(matrix1, rows, cols);
            break;
case 6:
            printf("Matrix:\n");
inputMatrix(matrix1, rows, cols);
transposeMatrix(matrix1, rows, cols);
            break;
case 7:
            printf("Matrix:\n");
inputMatrix(matrix1, rows, cols);             if
(isSymmetricMatrix(matrix1, rows, cols)) {
printf("The matrix is symmetric.\n");
            } else {                printf("The matrix is not
symmetric.\n");
            }             break;
default:             printf("Invalid
choice.\n");             break;
    }

    return 0;
}
```

RESULT:


"C:\Users\B Venkatesh\Desktop\c programming\matrices2\bin\Debug\matrices2.exe"
Matrix Operations:
1. Addition
2. Subtraction
3. Multiplication
4. Sum of diagonal or non-diagonal elements
5. Sum of rows and columns
6. Transpose of matrix
7. Check if matrix is symmetric
Enter your choice: 1
Enter the number of rows in the matrices: 2
Enter the number of columns in the matrices: 2
Matrix 1:
Enter the elements of the matrix:
1
2
3
4
Matrix 2:
Enter the elements of the matrix:
2
4
5
6
Addition of matrices:
Matrix:
3 6
8 10
Process returned 0 (0x0) execution time : 22.223 s
Press any key to continue.