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

// Function to add two matrices

```
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");
  }
```

```
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' \parallel 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;
```

```
"C:\Users\B Venkatesh\Desktop\c programming\matrices2\bin\Debug\matrices2.exe"
Matrix Operations:

    Addition

Subtraction
Multiplication
4. Sum of diagonal or non-diagonal elements
5. Sum of rows and columns
6. Transpose of matrix
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:
Matrix 2:
Enter the elements of the matrix:
Addition of matrices:
Matrix:
3 6
8 10
Process returned 0 (0x0) execution time : 22.223 s
Press any key to continue.
```

```
al6/23 Exp-1.
al Write C or C++ program to do the following
I Pass the matrices as palameters
in Adolition substoaction
sum of sow & columns
My Hydication
 ilson of posseple/non-poinciple diagonal elements
is point the tourspose of a given motorix
vill symmetoic or not
soln1-
#include < stolio. h7
# define MAX_SZZE 100
 tri, [3525-XAM][3565-XAM] riotan. tri) riotantugni bior
                  (2100 this, sware
  pointf("Ent& the elements of the matrix: \n");
  for ( Int 120; 1 c rous; 1++) (
     for (int )=0; ix cou; i++) 2
         scanf ("ld", &matrix[][]);
void point Matoix (int matoix [MAX-SIZE] [MAX-SIZE], it so
                     int coust
     pontf ("Matrixin");
     fox (++ i = 0; i < 80,003; i++) {
      for (int )=0; ] < co 15; j++)[
       printf("o/od", notix[i][i]);
```

```
poshtf (. /v.);
void add Motorices (Int motorial [MAX-522E] [MAX-522E], int
                  matoise [MAX_512E] [MAX_512E], it soul, it coly
  Int soult [MAX_SIZE] [MAX-SIZE];
   for (inti =0; i coover; i++)[
    for (int j=0; j < 1011; j++) s
     result[i][] = modsix[[i][i] + modsix2[i][i];
   pointf("In");
void subtrout Motorices (ind neutrice) [MAX-52E][MAX-182E],
   } (llas tin, elvos tri. [322-4A/] (3212-AA/) existeur tri
   int ocsult [MAX_S22E] [MAX_S22E];
   for (intioo; knows; it+) {
   for (int ) = = ; j < col; j++) {
     rust [i][j] = matoix [[i][j] - matoix 2[j][j];
 (( n)) +thisq
void nulliphylatricu (int matoix/[MAX-1928][MAX-1928],
 int source, int coll, motorne [MAX_522][MAX-522],
 int souls , int colls }
  A (coust != 80 ms 2) 5
    beryt ( , E0200 ,);
   seturn;
    FUNDAY_SIZE][MAX_SIZE])
 for (int i = 0; ic sows!; it) ?
    for (int) =0; j (10/152; j++) {
```

```
Desolt [1][] = 0;
     for (Nt k > 0) k < colst; k++) {
        result [i][i] += matrix[[i][k] * matrix2[k][i];
3(5
    points (" m");
   void son Diagonal Aba Diagonal (int matrix[MAX-522F][MAX-522F]
               int sous, int choice) {
       in som = 0;
    # (charce == 'D' | charce = = 'd') {
       for (Int 1 =0; iconos; it+) {
        sum += notoix [i][i];
       postf ("sun of diagonal elements! "Idh", sum);
      else f (choice == 'N' 11 choice == 'n'){
         for (intion, ic souss, i++){
           sum += motoix[i][j];
       postf ("sum of diagonal elements: "/d/n", sum);
      else if (charce 2= 'N' 11 charce = 2 'n') {
        loo(int 100; icoows; i++) {
          for (mt j=0; j < coll; j++) {
             if (i 1= j);
            sum+= motoix[][];
       printf ( 'non-diagonal elements 1 '/d/n', som);
```

```
point ("Invalid choice, please exter DOSN/n");
3 elec f
void transpose Matrix (int matrix [MAX-522E] [MAX-522E]
   } ( 21 a tri , 2000 tri
  int transposed [MAX-522E] [MAX-522E];
  for (it 120; 12000); 1+1){
   for (int )=0; jews; j++){
      transposed [i][i] = motrix[i][i];
   point (" In ")"
int symmetric Matorn (int matoix [MX_SEZE][MX_SEZE],
               } (2100 tri, 20000 tri
    if (800) 1= cou) {
      return 0;
  8 (Ht; 200; 1 < 2004); i++) }
      too (int )00; ) < cols; j++) {
      [i][i] xistom =! [i][i] xistom) Fi
        seturn o!
 oction 1;
```

) () main to ind choice, positif ("I. Addition 2. Substantion 3. Multiplaction 4. Sim of diagonal elements &. Story socret & S. Toanspose notoise 6. is symmetoise); point ("End your choice"); scanf (" · fod ", & choice); (llas, levos tris postf ("enter the number of sours"); scanf ("1.01", & sous); pointf("enter the number of columns"); scanf (" /d" & cols); ([3512-XAM][3512-XAM]Leistan ti int mustoige [MAX-SDE] [MAX-JDEE]; switch (choice) s case 1: input Motoix (motoix 1, sous, cous); input Matoise (matoises, souss, cols); and Matrices (matrial, matrixe, sous, u boeak, case 21 input matoix (matoix), sous, cols), input matrix (matrixe, sous, cols); substract Matrices (matrix), matrix2, rows boeak, coste 3! input matrix (matrix), some, coust, input matoix (matoix), saws, cols), multiply motorices (motorial, sours, cou, matrix2, col, sous); bosak)

```
care 4: input matrix (matrix), rous, cous;
       point ("Enter") for oliagonal elements out "N"
       for non-diagonals elements? ");
       chal som Choice;
      scant (" -/. ( ", & sumchaice);
      sumDiagonalNonDiagonal (natoix), 5000s, col,
           sum choice);
     break!
e S: input Motsix (matrix), sours, cols);
      sunkous (dumy (matoix), rows, cols);
      boeak!
e 6: inputhatoix (matrial, cols, sous);
      transpose protoix (notoix1, rous, cols),
     boeak!
  (2) ou, sours, Loiotan) sistentingni! [
      it (is symmetria hatria (notorial, sours, cols)) {
      point("Symmetrix");
    & elsef
      point ("non-symmetrix");
    boeak,
efault 1 points ("Invalid choice");
return
```