

Write a C program for the following

Pass matrices as parameters

1. Matrix addition and subtraction
2. Multiplication
3. Sum of Rows and Columns
4. Sum of principle diagonal and non principle diagonal
5. Print transpose of given matrix
6. Check if matrix is symmetric or not

Code:

```
#include<stdio.h>
#include<conio.h>
void add(int matrix1[3][3], int matrix2[3][3])
{
    int sum[3][3];
    for(int i=0;i<3;i++)
        for(int j=0;j<3;j++)
            sum[i][j]=matrix1[i][j]+matrix2[i][j];
    for(int i=0;i<3;i++){
        printf("\n");
        for(int j=0;j<3;j++){
            printf("\t%d",sum[i][j])
        }
    }
}

void subtract(int matrix1[3][3], int matrix2[3][3])
{
    int diff[3][3];
    for(int i=0;i<3;i++)
        for(int j=0;j<3;j++)
            diff[i][j]=matrix1[i][j]-matrix2[i][j];
    for(int i=0;i<3;i++){
        printf("\n");
        for(int j=0;j<3;j++){
            printf("\t%d",diff[i][j]);
        }
    }
}

void multiply(int matrix1[3][3], int matrix2[3][3])
{
    int product[3][3];
```

```

for(int i=0;i<3;i++){
    for(int j=0;j<3;j++){
        product[i][j]=0;
        {
            for(int k=0;k<3;k++){
                product[i][j]+=matrix1[i][k]*matrix2[k][j];
            }
        }
    }
}
for(int i=0;i<3;i++){
    printf("\n");
    for(int j=0;j<3;j++){
        printf("\t%d",product[i][j]);
    }
}
}

void sumOfRowsColumns(int matrix[3][3])
{
    int row_sum[3][4],column_sum[3][4],rowsum,columnsum;
    for(int i=0;i<3;i++){
        rowsum=0,columnsum=0;
        for(int j=0;j<3;j++){
            rowsum+=matrix[i][j];
            columnsum+=matrix[j][i];
        }
        row_sum[i][4]=rowsum;
        column_sum[i][4]=columnsum;
    }
    for(int i=0;i<3;i++){
        printf("\n");
        for(int j=0;j<4;j++){
            printf("\t%d",row_sum[i][j]);
        }
    }
    for(int i=0;i<3;i++){
        printf("\n");
        for(int j=0;j<4;j++){
            printf("\t%d",column_sum[i][j]);
        }
    }
}

```

```
}
```

```
void transpose(int matrix[3][3])
{
    int transpose[3][3];
    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            transpose[i][j]=matrix[j][i];
        }
    }
    for(int i=0;i<3;i++){
        printf("\n");
        for(int j=0;j<3;j++){
            printf("\t%d",transpose[i][j]);
        }
    }
}
```

```
void checkSymmetric(int matrix[3][3])
{
    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            if(matrix[i][j]!=matrix[j][i]){
                printf("\nAsymmetric matrix");
                return;
            }
        }
    }
    printf("\nMatrix is symmetric");
}
```

```
void sumOfDiagonals(int matrix[3][3])
{
    int sum=0,a=0;
    for (int i=0;i<3;++i) {

        sum = sum + matrix[i][i];
        a = a + matrix[i][3 - i - 1];
    }

    printf("\nMain diagonal elements sum is = %d\n", sum);
    printf("Off-diagonal elements sum is = %d\n", a);
}
```

}

Output:

```
Enter your choice: 3
Enter the number of rows and columns of matrix1: 2 2
Enter the number of columns of matrix2: 2 2
Enter elements of matrix1:
5
7
6
Enter elements of matrix2:
8
13
76
99
Resultant matrix after multiplication:
396    521
512    685

Matrix Operations:
1. Addition
2. Subtraction
3. Multiplication
4. Sum of Diagonals
5. Sum of Rows and Columns
6. Transpose
7. Check Symmetry
8. Exit
Enter your choice: 4
Enter the number of rows and columns of the matrix: 2 2
Enter elements of the matrix:
54
76
8
1
Sum of principal diagonal: 55
Sum of non-principal diagonal: 76

Matrix Operations:
1. Addition
2. Subtraction
3. Multiplication
4. Sum of Diagonals
5. Sum of Rows and Columns
6. Transpose
7. Check Symmetry
8. Exit
Enter your choice: 5
Enter the number of rows and columns of the matrix: 2 2
Enter elements of the matrix:
78
98
8
8
Sum of elements in Row 1: 168
Sum of elements in Row 2: 8
Sum of elements in Column 1: 78
Sum of elements in Column 2: 98
```

```
Matrix Operations:
1. Addition
2. Subtraction
3. Multiplication
4. Sum of Diagonals
5. Sum of Rows and Columns
6. Transpose
7. Check Symmetry
8. Exit
Enter your choice: 1
Enter the number of rows and columns of the matrices: 2 2
Enter elements of matrix1:
4
7
8
3
Enter elements of matrix2:
12
78
55
23
Resultant matrix after addition:
16      85
63      26

Matrix Operations:
1. Addition
2. Subtraction
3. Multiplication
4. Sum of Diagonals
5. Sum of Rows and Columns
6. Transpose
7. Check Symmetry
8. Exit
Enter your choice: 2
Enter the number of rows and columns of the matrices: 2 2
Enter elements of matrix1:
34
78
99
24
Enter elements of matrix2:
65
55
88
11
Resultant matrix after subtraction:
-31      23
11       13
```

```
Enter your choice: 6
Enter the number of rows and columns of the matrix: 2 2
Enter elements of the matrix:
1
43
56
7
Transpose of the matrix:
1      56
43     7

Matrix Operations:
1. Addition
2. Subtraction
3. Multiplication
4. Sum of Diagonals
5. Sum of Rows and Columns
6. Transpose
7. Check Symmetry
0. Exit
Enter your choice: 7
Enter the number of rows and columns of the matrix: 2 2
Enter elements of the matrix:
15
87
5
3
The matrix is not symmetric.
```

Observation:

Remarks

1) Write a C program for the following operations on matrices (14/06/23)

- 1) Addition Subtraction
- 2) Matrix multiplication
- 3) Sum of principal & non principal diagonals
- 4) Sum of rows & columns
- 5) Print a transpose of given matrix
- 6) Check if a given matrix is symmetric or not

Sol) #include <stdio.h>

```
void printMatrix(int r, int c, int matrix[r][c]) {
    for(int i=0; i<r; i++) {
        for(int j=0; j<c; j++) {
            printf("%d\t", matrix[i][j]);
        }
        printf("\n");
    }
}
```

```
void matrixAddition(int r, int c, int matrix1[r][c], int matrix2[r][c]) {
    int result[r][c];
```

```
    for(int i=0; i<r; i++) {
        for(int j=0; j<c; j++) {
            result[i][j] = matrix1[i][j] + matrix2[i][j];
        }
    }
```

```
    printf("Matrix Addition: \n");
    printMatrix(r, c, result);
}
```

```

void matrixSubtraction (int n, int c, int matrix1[c][c],
int matrix2[c][c]) {
    int result[c][c];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < c; j++) {
            result[i][j] = matrix1[i][j] - matrix2[i][j];
        }
    }
}

// Example: matrixSubtraction (matrix1, matrix2);
// matrixSubtraction (matrix1, matrix2);

void matrixMultiplication (int n1, int c1, int matrix1[c1][c1],
int n2, int c2, int matrix2[c2][c2]) {
    if (c1 != n2) {
        printf("Invalid");
        return;
    }
    int result[n1][c2];
    for (int i = 0; i < n1; i++) {
        for (int j = 0; j < c2; j++) {
            result[i][j] = 0;
            for (int k = 0; k < c1; k++) {
                result[i][j] += matrix1[i][k] * matrix2[k][j];
            }
        }
    }
}

```



int matrix[10][10],

int matrix[10][10],

int matrix[10][10],

int matrix[10][10],

int matrix[10][10],

int matrix[10][10],

Print the Addition matrix.

Print matrix (size, col, row), 5

int sumPairWise (int size, int matrix[10][10]) {

int sum=0;

for (int i=0; i<size; i++)

sum += matrix[i][i];

return sum;

}

int sumNewPairWise (int size, int matrix[10][10]) {

int sum=0;

for (int i=0; i<size; i++)

sum += matrix[i][size-i-1];

return sum;

}

int sumRow (int row, int column, int matrix[10][10]) {

int sum=0;

for (int i=0; i<column; i++)

sum += matrix[row][i];

return sum;

}

int sumColumn (int row, int column, int matrix[10][10]) {

int sum=0;

for (int i=0; i<row; i++)

sum += matrix[i][column]; return sum;

}

```

int isSymmetric(int size, int matrix[size][size]) {
    for (int i = 0; i < size; i++) {
        for (int j = i + 1; j < size; j++) {
            if (matrix[i][j] != matrix[j][i]) {
                return 0;
            }
        }
    }
    return 1;
}

int main() {
    int size;
    Print("Enter size of rows & columns:");
    scanf("%d", &size);
    int matrix[size][size], i, j;
    printf("Enter the matrix\n");
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }
}

```



Case 4:

```
printf("Sum of principal diagonal: %d\n",  
sumPrincipalDiagonal(row, matrix));
```

break;

Case 5:

```
printf("Sum of main principal diagonal: %d\n", sumMainPrincipal  
(row, matrix));
```

break;

Case 6:

```
printf("Enter row index (0-%d): ", row-1);
```

int row;

```
scanf("%d", &row);
```

if (row >= 0 && row < row1)

```
printf("Sum of row %d: %d\n", row, sumRow(row,  
C, matrix));
```

else

```
printf("Invalid row index\n");
```

else

break;

Case 7:

```
printf("Enter column index (0-%d): ", column-1);
```

int column;

```
scanf("%d", &column);
```

if (column >= 0 && column < column1)

3. the S  
 (n) V ("invalid column index")

3  
hwa 10

Proof: "Show part of the result. (1)";  
next assume (2) & (3) and deduce (1);

```
print modinv(2, C, modulus);
```

16 (is Symmetric (2, 1) matrix))

phospholipid model is symmetrical

But  $\Gamma$  moduli is not symmetric. (11.1)

philtre machine is not so small as it

with  $(n \text{ expts. } \dots n!)$

$\text{mit } f(x) = \frac{1}{x^2}$

default:  $\frac{1}{2} \ln \frac{1}{2}$

Phis / (I inward) Chosen - In 1

3.  $\frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$

Ex 16. 2. 2nd ed.

$$B = \begin{pmatrix} 1 & 2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$A-B = \begin{bmatrix} 0 & 2 & 3 \\ 4 & 0 & 1 \\ 7 & 6 & 8 \end{bmatrix}$$

$$A^* B = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

प्रमाणों के हैं ✓

It is significant

of dissolved

Sum of non diagonal elements

Sum of how q,  $\Gamma = \sqrt{1 - 9}$

1	2	3	6
9	5	6	15
7	8	9	

Som of column of A :-

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 7 & 6 \\ 7 & 8 & 9 \\ 12 & 15 & 18 \end{bmatrix}$$

✓  
21/6/23

✓

