

```
#include <stdio.h>

#define MAX_SIZE 10

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

void addMatrices(int matrix1[][MAX_SIZE], int
matrix2[][MAX_SIZE], int result[][MAX_SIZE], int size) {
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            result[i][j] = matrix1[i][j] + matrix2[i][j];
        }
    }
}

void subtractMatrices(int matrix1[][MAX_SIZE], int
matrix2[][MAX_SIZE], int result[][MAX_SIZE], int size) {
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            result[i][j] = matrix1[i][j] - matrix2[i][j];
        }
    }
}

void multiplyMatrices(int matrix1[][MAX_SIZE], int
matrix2[][MAX_SIZE], int result[][MAX_SIZE], int size) {
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            result[i][j] = 0;
            for (int k = 0; k < size; k++) {
```

```

        result[i][j] += matrix1[i][k] *
matrix2[k][j];
    }
}

}

}

int sumPrincipalDiagonal(int matrix[][MAX_SIZE], int size) {
    int sum = 0;
    for (int i = 0; i < size; i++) {
        sum += matrix[i][i];
    }
    return sum;
}

int sumNonPrincipalDiagonal(int matrix[][MAX_SIZE], int
size) {
    int sum = 0;
    for (int i = 0; i < size; i++) {
        sum += matrix[i][size - i - 1];
    }
    return sum;
}

void sumRows(int matrix[][MAX_SIZE], int size) {
    for (int i = 0; i < size; i++) {
        int sum = 0;
        for (int j = 0; j < size; j++) {
            sum += matrix[i][j];
        }
        printf("Sum of elements in row %d: %d\n", i + 1,
sum);
    }
}

void sumColumns(int matrix[][MAX_SIZE], int size) {
    for (int i = 0; i < size; i++) {
        int sum = 0;

```

```

        for (int j = 0; j < size; j++) {
            sum += matrix[j][i];
        }
        printf("Sum of elements in column %d: %d\n", i + 1,
sum);
    }
}

void transposeMatrix(int matrix[][MAX_SIZE], int size) {
    int temp;
    for (int i = 0; i < size; i++) {
        for (int j = i + 1; j < size; j++) {
            temp = matrix[i][j];
            matrix[i][j] = matrix[j][i];
            matrix[j][i] = temp;
        }
    }
}

int isSymmetric(int matrix[][MAX_SIZE], int 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 matrix1[MAX_SIZE][MAX_SIZE];
    int matrix2[MAX_SIZE][MAX_SIZE];
    int result[MAX_SIZE][MAX_SIZE];

    int size;
    printf("Enter the size of the square matrices: ");
    scanf("%d", &size);

```

```
printf("Enter the elements of Matrix 1:\n");
for (int i = 0; i < size; i++) {
    for (int j = 0; j < size; j++) {
        scanf("%d", &matrix1[i][j]);
    }
}

printf("Enter the elements of Matrix 2:\n");
for (int i = 0; i < size; i++) {
    for (int j = 0; j < size; j++) {
        scanf("%d", &matrix2[i][j]);
    }
}

int choice;
int exitFlag = 0;

while (!exitFlag) {
    printf("\nMatrix Operations\n");
    printf("1. Add Matrices\n");
    printf("2. Subtract Matrices\n");
    printf("3. Multiply Matrices\n");
    printf("4. Sum of Principal Diagonal\n");
    printf("5. Sum of Non-Principal Diagonal\n");
    printf("6. Sum of Rows\n");
    printf("7. Sum of Columns\n");
    printf("8. Print Transpose\n");
    printf("9. Check Symmetry\n");
    printf("0. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch (choice) {
        case 0:
            exitFlag = 1;
            break;
        case 1:
```

```

        printf("Matrix 1:\n");
        printMatrix(matrix1, size);

        printf("\nMatrix 2:\n");
        printMatrix(matrix2, size);

        printf("\nAdding matrices:\n");
        addMatrices(matrix1, matrix2, result, size);
        printMatrix(result, size);
        break;
    case 2:
        printf("Matrix 1:\n");
        printMatrix(matrix1, size);

        printf("\nMatrix 2:\n");
        printMatrix(matrix2, size);

        printf("\nSubtracting matrices:\n");
        subtractMatrices(matrix1, matrix2, result,
size);

        printMatrix(result, size);
        break;
    case 3:
        printf("Matrix 1:\n");
        printMatrix(matrix1, size);

        printf("\nMatrix 2:\n");
        printMatrix(matrix2, size);

        printf("\nMultiplying matrices:\n");
        multiplyMatrices(matrix1, matrix2, result,
size);

        printMatrix(result, size);
        break;
    case 4:
        printf("Matrix:\n");
        printMatrix(matrix1, size);

```

```
        printf("\nSum of Principal Diagonal: %d\n",
sumPrincipalDiagonal(matrix1, size));
        break;
    case 5:
        printf("Matrix:\n");
        printMatrix(matrix1, size);

        printf("\nSum of Non-Principal Diagonal:
%d\n", sumNonPrincipalDiagonal(matrix1, size));
        break;
    case 6:
        printf("Matrix:\n");
        printMatrix(matrix1, size);

        printf("\nSum of Rows:\n");
        sumRows(matrix1, size);
        break;
    case 7:
        printf("Matrix:\n");
        printMatrix(matrix1, size);

        printf("\nSum of Columns:\n");
        sumColumns(matrix1, size);
        break;
    case 8:
        printf("Matrix:\n");
        printMatrix(matrix1, size);

        printf("\nTranspose of Matrix:\n");
        transposeMatrix(matrix1, size);
        printMatrix(matrix1, size);
        break;
    case 9:
        printf("Matrix:\n");
        printMatrix(matrix1, size);

        if (isSymmetric(matrix1, size)) {
            printf("\nThe matrix is symmetric.\n");
```

```
        } else {  
            printf("\nThe matrix is not  
symmetric.\n");  
        }  
        break;  
    default:  
        printf("Invalid choice.\n");  
        break;  
    }  
}  
  
return 0;  
}
```