```
#include<stdio.h>
void addMatrix(int mat1[][100], int mat2[][100],
 int result[][100], int rows, int cols)
    {for(int i=0;i<rows;i++)
        {for(int j=0;j<cols;j++)
            {result[i][j]=mat1[i][j]+mat2[i][j];
            }
        }
    }
void subtractMatrix(int mat1[][100], int mat2[][100],
    int result[][100], int rows, int cols)
    {for(int i=0;i<rows;i++)
        {for(int j=0;j<cols;j++)
            {result[i][j]=mat1[i][j]-mat2[i][j];
            }
        }
    }
void multiplyMatrix(int mat1[][100], int mat2[][100],
int result[][100], int rows1, int cols1, int cols2)
{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]+=mat1[i][k]*mat2[k][j];
            }
        }
    }
}
```

```
void diagonalSum(int matrix[][10], int size)
{ int principalSum = 0, nonPrincipalSum = 0;
  for(int i=0; i<size; i++)
  { principalSum += matrix[i][i];
    nonPrincipalSum += matrix[i][size-1-i];
  }
}
void rowColumnSum(int matrix[][10], int rows,
                  int cols)
{ for(int i=0; i<rows; i++)
  { int rowSum = 0;
    for(int j=0; j<cols; j++)
    { rowSum += matrix[i][j];
    }
    for(int j=0; j<cols; j++)
    { int colSum = 0;
      for(int i=0; i<rows; i++)
      { colSum += matrix[i][j];
      }
      printf("Sum of elements %d: %d\n", j+1,
             colSum);
    }
}
void printTranspose(int matrix[][10], int
                    rows, int cols)
{ for(int j=0; j<cols; j++)
  { for(int i=0; i<rows; i++)
    { printf("%d ", matrix[i][j]);
    }
    printf("\n");
  }
}
```

```
int isSymmetric ( int matrix [ ][100], 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
      { return 0
    } }
  return 1;
}

int main ()
{
case 1:
  for ( int i = 0 ; i < rows1 ; i++)
    { for ( int j = 0 ; j < cols1 ; j++)
  for ( int i = 0 ; i < rows1 ; i++)
    { for ( int j = 0 ; j < cols1 ; j++)
    }
add matrix
  for ( int i = 0 ; i < rows1 ; i++)
    { for ( int j = 0 ; j < cols1 ; j++)
```

case 2.

```
for (int i =0 ; i < rows1 ; i++)
  {for (int j =0 ; j < cols1 ; j++)
    { scanf ("%d", &matrix1[i][j])
```

subtractMatrix ()

```
for (int i =0; i < rows1; i++)
  { for (int j =0; j < cols1; j++)
    {
```

~~case 3~~:

multiplyMatrix ()

```
for (int i =0; i < rows1; i++)
  { for (int j =0; j < cols2; j++
```

diagonalsum (matrix1, &rows1);

~~b~~

~~Output~~

~~case 4~~:

case 3:

```
printf ("Enter the number of rows and
columns of matrix 1:");
scanf ("%d %d", &rows1, &cols1);
printf ("Enter the number of columns
& matrix2:");
scanf ("%d", &cols2);
printf ("Enter elements of matrix 1:")
  for (int i=0 ; i < rows1; i++)
    { for (int j=0; j < cols1; j++)
```

```
        { scanf ("%d", &matrix1[i][j]);
        }
    }
    printf ("Enter elements of matrix 2:\n");
    for (int i=0; i<cols1; i++)
        { for (int j=0; j<cols2; j++)
            {scanf ("%d", &matrix2[i][j]);
        }
    }
    multiplyMatrix (matrix1, matrix2,
    result, rows1, cols1, cols2);
    printf ("Resultant matrix after
    multiplication:");
    for (int i=0; i<rows1; i++)
        { for (int j=0; j<cols2; j++)
            {Print| ("%d ", result[i][j]);
        }
        printf ("\n");
    }
    break;

case 4:
printf ("Enter the number of rows and
columns of the matrix:");
scanf ("%d %d", &rows1, &cols1);
printf ("Enter elements of matrix:");
    for (int i=0; i<rows1; i++)
        { for (int j=0; j<cols1; j++)
            { scanf ("%d", &matrix1[i][j]);
}
```

```
diagonalSum (matrix1, rows1);
    break;

case 5:
    printf ("Enter the number of rows and
columns of the matrix:");
    scanf ("%d %d", &rows1, &cols1);
    printf ("Enter the elements of matrix:");
    for(int i=0; i<rows1; i++)
    {for(int j=0; j<cols1; j++)
    {scanf ("%d", &matrix1[i][j]);
    }}
    rowsColumnSum (matrix1, rows1, cols1)
    break;

case 6:
printf ("Enter the number of rows and
columns of the matrix:");
    scanf ("%d%d", &rows1, &cols1);
    printf ("Enter elements of the matrix:");
for(int i=0; i<rows1; i++)
    {for (int j=0; j<cols1; j++)
        {scanf ("%d", &matrix1[i][j]);
    }}
Printf Transpose (matrix1, rows1, cols1);
    break;
}
```

case 7:
printf ("Enter the number of rows and columns of the matrix:");
scanf ("%d %d", &rows1, &cols1);
printf ("Enter elements of the matrix:");
for (int i=0; i<rows1; i++)
{ for (int j=0; j<cols1; j++)
{ scanf ("%d", &matrix1[i][j]);
}
if (isSymmetric (matrix1, rows1, cols1))
{ printf ("The matrix is symmetric:\n");
}
break;
case 0:
printf ("Exiting the program.\n");
Goodbye!\n");
break;
default:
printf ("Invalid choice. Please enter a valid option\n");
}
printf ("\n");
} while (choice != 0);
return 0;
}

Output

Matrix operat
1) A
2) Subtra
3) multipli
4) Sum of
5) Sum of
6) Transpo
7) Check s
0 Exit
Enter you
Enter num
Enter el
4
7
8
3
Enter el
12
28
55
23
Resultant
16 85
63 26

Enter cho
Enter ele
34
78
99
24
Enter ele
65
55

```
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
0. Exit
Enter your choice: 4
Enter the number of rows and columns of the matrix: 2 2
Enter elements of the matrix:
54
76
0
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
0. Exit
Enter your choice: 5
Enter the number of rows and columns of the matrix: 2 2
Enter elements of the matrix:
78
90
0
0
Sum of elements in Row 1: 168
Sum of elements in Row 2: 0
Sum of elements in Column 1: 78
Sum of elements in Column 2: 90
```

```
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: 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
0. 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.
```