Lab 1

Write a program to pass matrices as parameters in the program to perform the following:

- Addition and subtraction
- Sum of diagonal elements
- Sum of rows and coloumns
- Transpose
- Check whether the matrix is symmetric

CODE:

```
#include <stdio.h>

#include <conio.h>

void add(int a[10][10], int b[10][10], int sum[10][10], int r, int c)
{
   for (int i = 0; i < r; i++)
   {
        for (int j = 0; j < c; j++)
       {
         sum[i][j] = a[i][j] + b[i][j];
      }
   }
}
void diff(int a[10][10], int b[10][10], int diff[10][10], int r, int c)
{
   for (int i = 0; i < r; i++)
   {
        for (int j = 0; j < c; j++)
       {
         diff[i][j] = a[i][j] - b[i][j];
      }
   }
}

void multiply(int a[10][10], int b[10][10], int mul[10][10], int r, int c)
{
   for (int i = 0; i < r; i++)
   {
        for (int j = 0; j < c; j++)
```

```
          {
           mul[i][j] = 0;
           for (int k = 0; k < c; k++)
           {
             mul[i][j] += a[i][k] * b[k][j];
           }
        }
    }
}
void rc_sum(int a[10][10], int r, int c)
{
    int rsum, csum;

    printf("Row sum:\n");
    for (int i = 0; i < r; i++) {
        rsum= 0;
        for (int j = 0; j < c; j++) {
            rsum += a[i][j];
        }
        printf("Row %d: %d\n", i + 1, rsum);
    }

    printf("Coloumn sum:\n");
    for (int j = 0; j < columns; j++) {
        csum = 0;
        for (int i = 0; i < r; i++) {
            csum += a[i][j];
        }
        printf("Column %d: %d\n", j + 1, csum);
    }
}

void transpose(int a[10][10], int r, int c) {
    int trans[10][10];

    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            trans[j][i] = a[i][j];
        }
    }

    printf("Transpose Matrix\n");
    for (int i = 0; i < c; i++)
    {
        for (int j = 0; j < r; j++)
         {
```

```
            printf("%d", trans[i][j]);
        }
    }
}
int isSymmetric(int a[10][10], int r, int c) {
    if ( r!= c)
        {
        return 0;
       }

    for (int i = 0; i < r; i++)
    {
        for (int j = 0; j < c; j++)
        {
            if (a[i][j] != b[j][i])
             {
                return 0;
            }
        }
    }

    return 1;
}


int main() {
    int matrix1[100][100], matrix2[100][100], result[100][100];
    int rows1, cols1, rows2, cols2;
    int choice;

    do {
        printf("Matrix Operations:\n");
        printf("1. Addition\n");
        printf("2. Subtraction\n");
        printf("3. Multiplication\n");
        printf("4. Sum of Diagonals\n");
        printf("5. Sum of Rows and Columns\n");
        printf("6. Transpose\n");
        printf("7. Check Symmetry\n");
        printf("0. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter the number of rows and columns of the matrices: ");
```

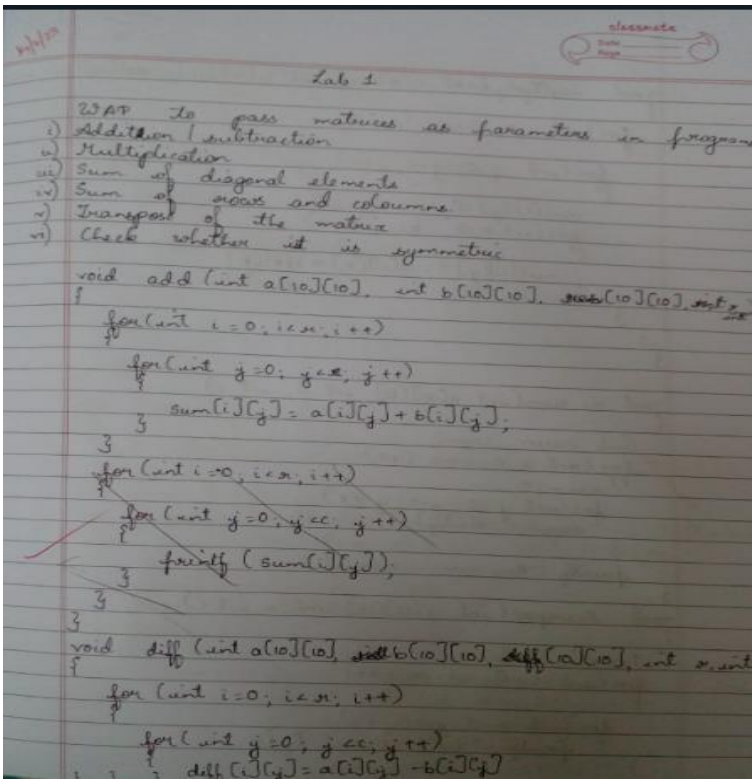
```
    scanf("%d %d", &rows1, &cols1);

    printf("Enter elements of matrix1:\n");
    for (int i = 0; i < rows1; i++) {
        for (int j = 0; j < cols1; j++) {
            scanf("%d", &matrix1[i][j]);
        }
    }

    printf("Enter elements of matrix2:\n");
    for (int i = 0; i < rows1; i++) {
        for (int j = 0; j < cols1; j++) {
            scanf("%d", &matrix2[i][j]);
        }
    }

    addMatrix(matrix1, matrix2, result, rows1, cols1);

    printf("Resultant matrix after addition:\n");
    for (int i = 0; i < rows1; i++) {
        for (int j = 0; j < cols1; j++) {
            printf("%d\t", result[i][j]);
        }
        printf("\n");
    }
    break;

case 2:
    printf("Enter the number of rows and columns of the matrices: ");
    scanf("%d %d", &rows1, &cols1);

    printf("Enter elements of matrix1:\n");
    for (int i = 0; i < rows1; i++) {
        for (int j = 0; j < cols1; j++) {
            scanf("%d", &matrix1[i][j]);
        }
    }

    printf("Enter elements of matrix2:\n");
    for (int i = 0; i < rows1; i++) {
        for (int j = 0; j < cols1; j++) {
            scanf("%d", &matrix2[i][j]);
        }
    }

    subtractMatrix(matrix1, matrix2, result, rows1, cols1);
```

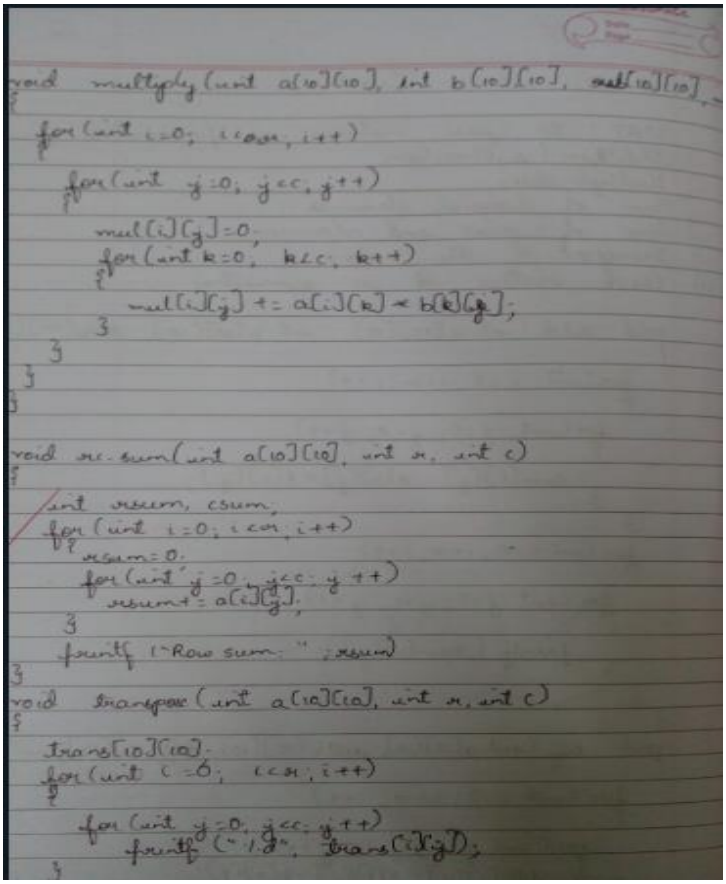
```
    printf("Resultant matrix after subtraction:\n");
    for (int i = 0; i < rows1; i++) {
        for (int j = 0; j < cols1; j++) {
            printf("%d\t", result[i][j]);
        }
        printf("\n");
    }
    break;

case 3:
    printf("Enter the number of rows and columns of matrix1: ");
    scanf("%d %d", &rows1, &cols1);
    printf("Enter the number of columns of matrix2: ");
    scanf("%d", &cols2);

    printf("Enter elements of matrix1:\n");
    for (int i = 0; i < rows1; i++) {
        for (int j = 0; j < cols1; j++) {
            scanf("%d", &matrix1[i][j]);
        }
    }

    printf("Enter elements of matrix2:\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:\n");
    for (int i = 0; i < rows1; i++) {
        for (int j = 0; j < cols2; j++) {
            printf("%d\t", 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 the matrix:\n");
```

```
    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 elements of the matrix:\n");
    for (int i = 0; i < rows1; i++) {
        for (int j = 0; j < cols1; j++) {
            scanf("%d", &matrix1[i][j]);
        }
    }

    rowColumnSum(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:\n");
    for (int i = 0; i < rows1; i++) {
        for (int j = 0; j < cols1; j++) {
            scanf("%d", &matrix1[i][j]);
        }
    }

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

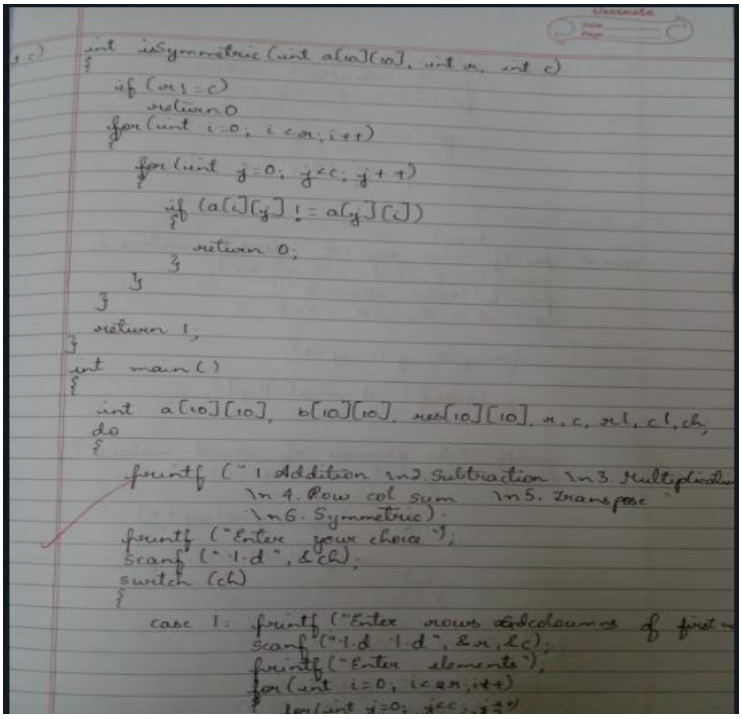
```
            }

            if (isSymmetric(matrix1, rows1, cols1)) {
                printf("The matrix is symmetric.\n");
            } else {
                printf("The matrix is not symmetric.\n");
            }
            break;

        case 0:
            printf("Exiting the program. Goodbye!\n");
            break;

        default:
            printf("Invalid choice. Please enter a valid option.\n");
    }

    printf("\n");

  } while (choice != 0);

  return 0;
}
```

OBSERVATION BOOK :

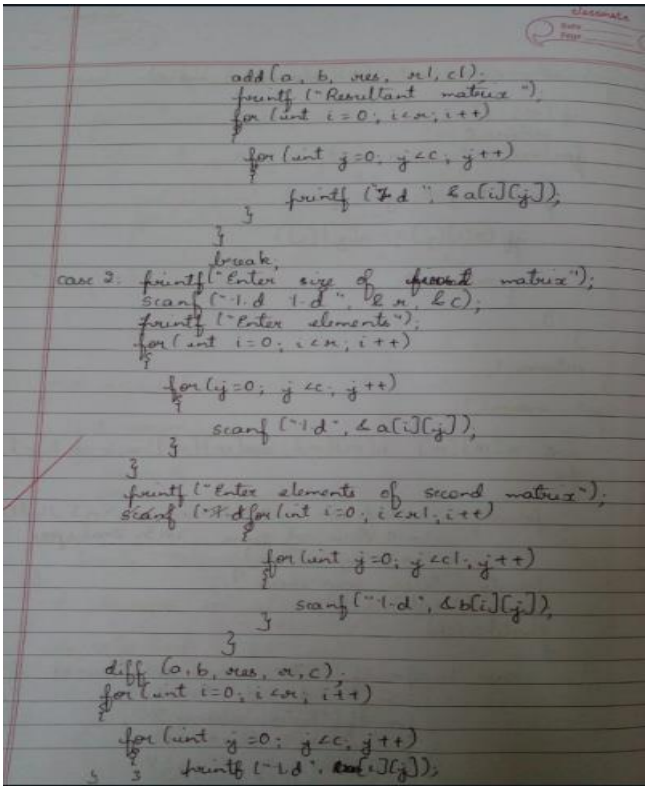




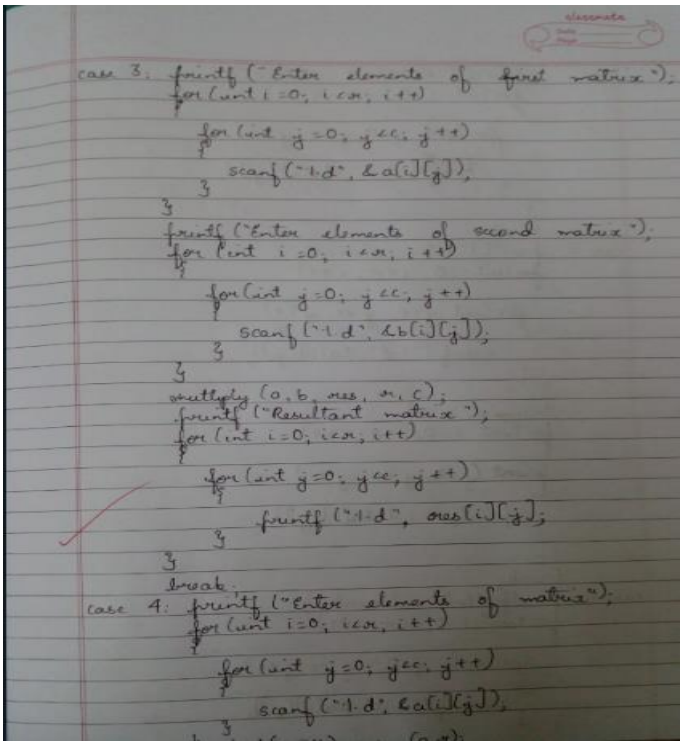
```
void multiply(int a[10][10], int b[10][10], mul[10][10])
{
  for(int i=0; i<r; i++)
    for(int j=0; j<c; j++)
      mul[i][j]=0;
      for(int k=0; k<c; k++)
      {
        mul[i][j] += a[i][k] * b[k][j];
      }
  }
}

void rc_sum(int a[10][10], int r, int c)
{
  int rsum, csum;
  for(int i=0; i<r; i++)
    rsum=0;
    for(int j=0; j<c; j++)
      rsum+= a[i][j];
  }
  printf("Row sum: ", rsum)
}
void transpose(int a[10][10], int r, int c)
{
  trans[10][10];
  for(int i=0; i<r; i++)
  {
    for(int j=0; j<c; j++)
      printf("%d", trans[i][j]);
  }
```



```
int isSymmetric(int a[10][10], int r, int c)
{
  if(r!=c)
    return 0
  for(int i=0; i<r; i++)
  {
    for(int j=0; j<c; j++)
      if(a[i][j] != a[j][i])
      {
        return 0;
      }
  }
  return 1;
}
int main()
{
  int a[10][10], b[10][10], res[10][10], r, c, r1, c1, ch;
  do
  {
    printf("1 Addition 2n3 Subtraction 3n3 Multiplicat
            \n4 Row col sum 5n5 Transpose
            \n6 Symmetric");
    printf("Enter your choice");
    scanf("%d", &ch);
    switch(ch)
    {
      case 1: printf("Enter rows &columns of first
              scanf("%d %d", &r, &c);
              printf("Enter elements");
              for(int i=0; i<r; i++)
              for(int j=0; j<c; j++)
```



```
        add(a, b, res, r1, c1);
        printf("Resultant matrix");
        for(int i=0; i<r; i++)
        {
          for(int j=0; j<c; j++)
          {
            printf("%d ", &a[i][j]);
          }
        }
        break;
case 2: printf("Enter size of second matrix");
        scanf("%d %d", &r, &c);
        printf("Enter elements");
        for(int i=0; i<r; i++)
        {
          for(j=0; j<c; j++)
          {
            scanf("%d", &a[i][j]);
          }
        }
        printf("Enter elements of second matrix");
        scanf("%d for(int i=0; i<r1; i++)
              {
                for(int j=0; j<c1; j++)
                {
                  scanf("%d", &b[i][j]);
                }
              }
        diff(a, b, res, r, c);
        for(int i=0; i<r; i++)
        {
          for(int j=0; j<c; j++)
          printf("%d", [j]);
        }
```



```
case 3: printf("Enter elements of first matrix");
        for(int i=0; i<r; i++)
        {
          for(int j=0; j<c; j++)
          {
            scanf("%d", &a[i][j]);
          }
        }
        printf("Enter elements of second matrix");
        for(int i=0; i<r; i++)
        {
          for(int j=0; j<c; j++)
          {
            scanf("%d", &b[i][j]);
          }
        }
        multiply(a, b, res, r, c);
        printf("Resultant matrix");
        for(int i=0; i<r; i++)
        {
          for(int j=0; j<c; j++)
          {
            printf("%d", res[i][j]);
          }
        }
        break;
case 4: printf("Enter elements of matrix");
        for(int i=0; i<r; i++)
        {
          for(int j=0; j<c; j++)
          {
            scanf("%d", &a[i][j]);
          }
```

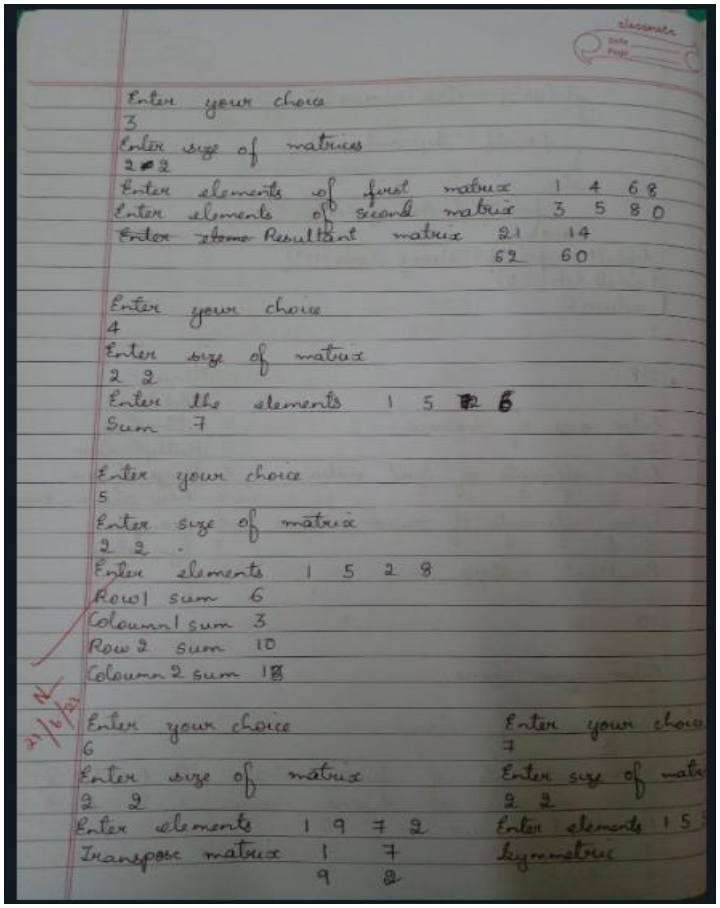
```
        printf("Resultant matrix");
        for(int i=0; i<r; i++)
        {
            for(int j=0; j<c; j++)
                printf("%d", a[i][j]);
        }
    }
    break;
case 5: printf("Enter elements");
        for(int i=0; i<r; i++)
        {
            for(int j=0; j<c; j++)
                scanf("%d", &a[i][j]);
        }
    }
    transpose(a, r, c);
    for(int i=0; i<r; i++)
    {
        for(int j=0; j<c; j++)
            printf("%d", a[i][j]);
        }
    }
    break;
case 6: printf("Enter elements");
        for(int i=0; i<r; i++)
        {
            for(int j=0; j<c; j++)
                scanf("%d", &a[i][j]);
        }
    }
```

```
    if(isSymmetric(a, r, c))
    {
        printf("Symmetric");
    }
    else
        printf("Not symmetric");
    break;
default: printf("Wrong choice!!!");
} while(ch!=0);
return 0;
}
```

O/P:

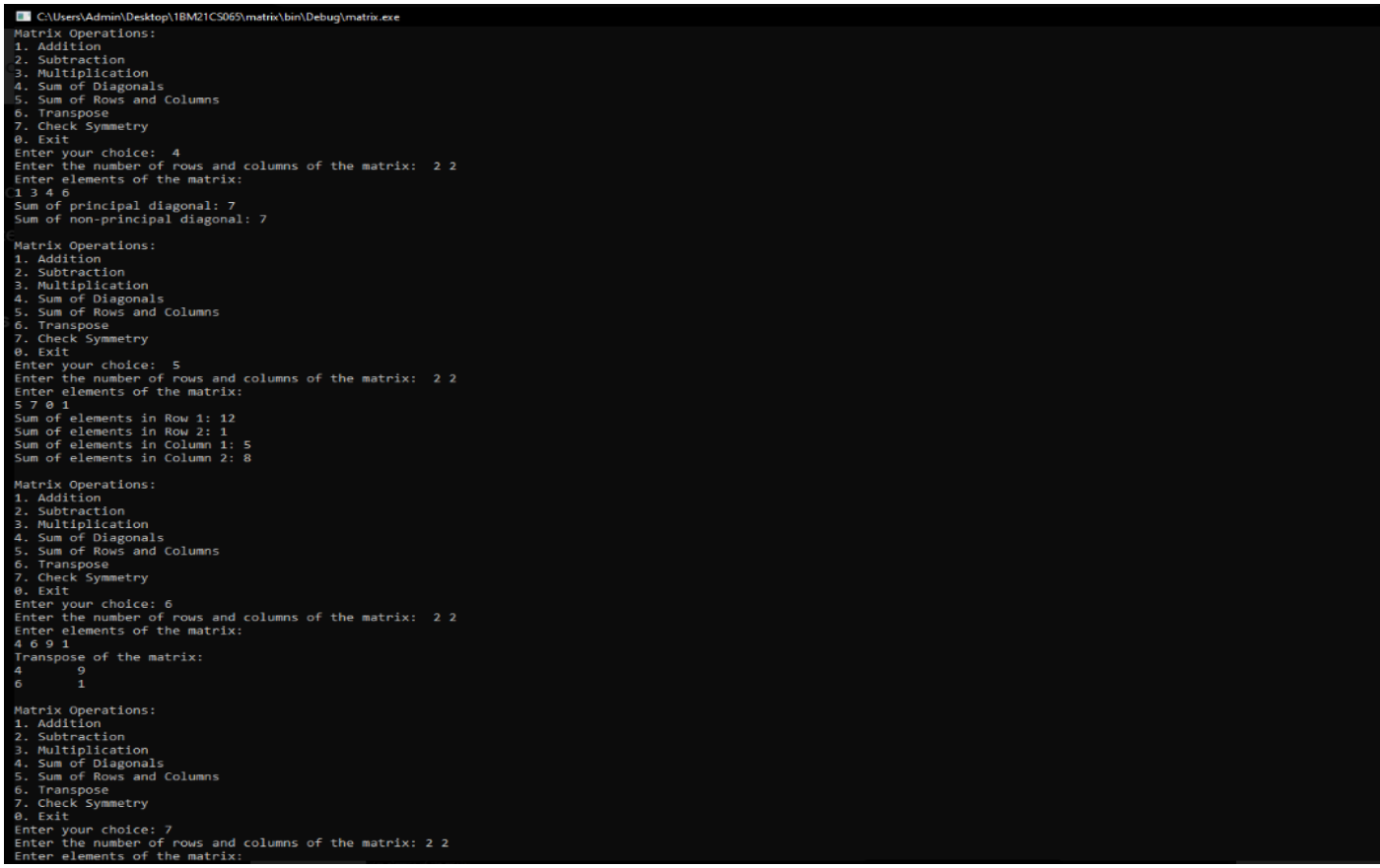
Menu:
1. Addition
2. Subtraction
3. Multiplication
4. Diagonal sum
5. Row column sum
6. Transpose
7. Symmetry

```
Enter size of matrices
2 2
Enter elements of first matrix
1 2 3 4
Enter elements of second matrix
5 6 7 8
Resultant matrix
6 8
10 12

Enter your choice
2
Enter size of matrices
2 2
Enter elements of first matrix 5 6 7 8
Enter elements of second matrix 1 2 3 4
Resultant matrix
4 4
4 4
```



OUTPUT:

```
C:\Users\Admin\Desktop\1BM21CS065\matrix\bin\Debug\matrix.exe
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:
1 2 3 4
Enter elements of matrix2:
5 6 7 8
Resultant matrix after addition:
6       8
10      12

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:
5 6 7 8
Enter elements of matrix2:
1 2 3 4
Resultant matrix after subtraction:
4       4
4       4
```


C:\Users\Admin\Desktop\1BM21CS065\matrix\bin\Debug\matrix.exe
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:
1 3 4 6
Sum of principal diagonal: 7
Sum of non-principal diagonal: 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: 5
Enter the number of rows and columns of the matrix: 2 2
Enter elements of the matrix:
5 7 0 1
Sum of elements in Row 1: 12
Sum of elements in Row 2: 1
Sum of elements in Column 1: 5
Sum of elements in Column 2: 8
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: 6
Enter the number of rows and columns of the matrix: 2 2
Enter elements of the matrix:
4 6 9 1
Transpose of the matrix:
4 9
6 1
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:


C:\Users\Admin\Desktop\1BM21CS065\matrix\bin\Debug\matrix.exe
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:
4 3 8 9
The matrix is not symmetric.
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: 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:
1 4 6 8
Enter elements of matrix2:
5 5 8 0
Resultant matrix after multiplication:
21 14
52 60
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: Exiting the program. Goodbye!
Process returned 0 (0x0) execution time : 69.229 s
Press any key to continue.