MATRIX

WACP to pass matrices as parameters in the following programs:
a)Addition
b)Subtraction
c)Multiplication
d)Sum of diagonal elements
e)Sum of rows and columns
f)Transpose
g)Check if symmetric or not

Code:

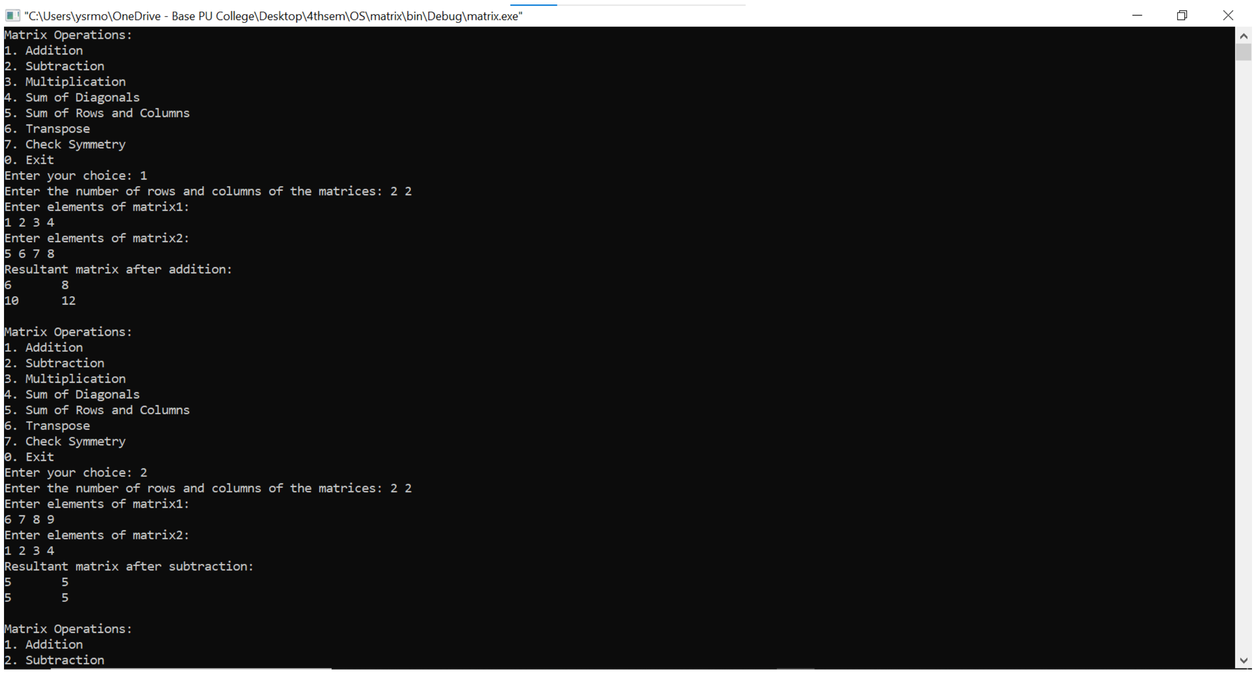
```
#include <stdio.h>

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[][100], int size) {
    int principalSum = 0, nonPrincipalSum = 0;

    for (int i = 0; i < size; i++) {
        principalSum += matrix[i][i];
        nonPrincipalSum += matrix[i][size - 1 - i];
    }

    printf("Sum of principal diagonal: %d\n", principalSum);
    printf("Sum of non-principal diagonal: %d\n", nonPrincipalSum);
}

void rowColumnSum(int matrix[][100], 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];
        }
        printf("Sum of elements in Row %d: %d\n", i + 1, rowSum);
    }

    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 in Column %d: %d\n", j + 1, colSum);
    }
}

void printTranspose(int matrix[][100], int rows, int cols) {
    printf("Transpose of the matrix:\n");
    for (int j = 0; j < cols; j++) {
        for (int i = 0; i < rows; i++) {
            printf("%d\t", 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][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);
```

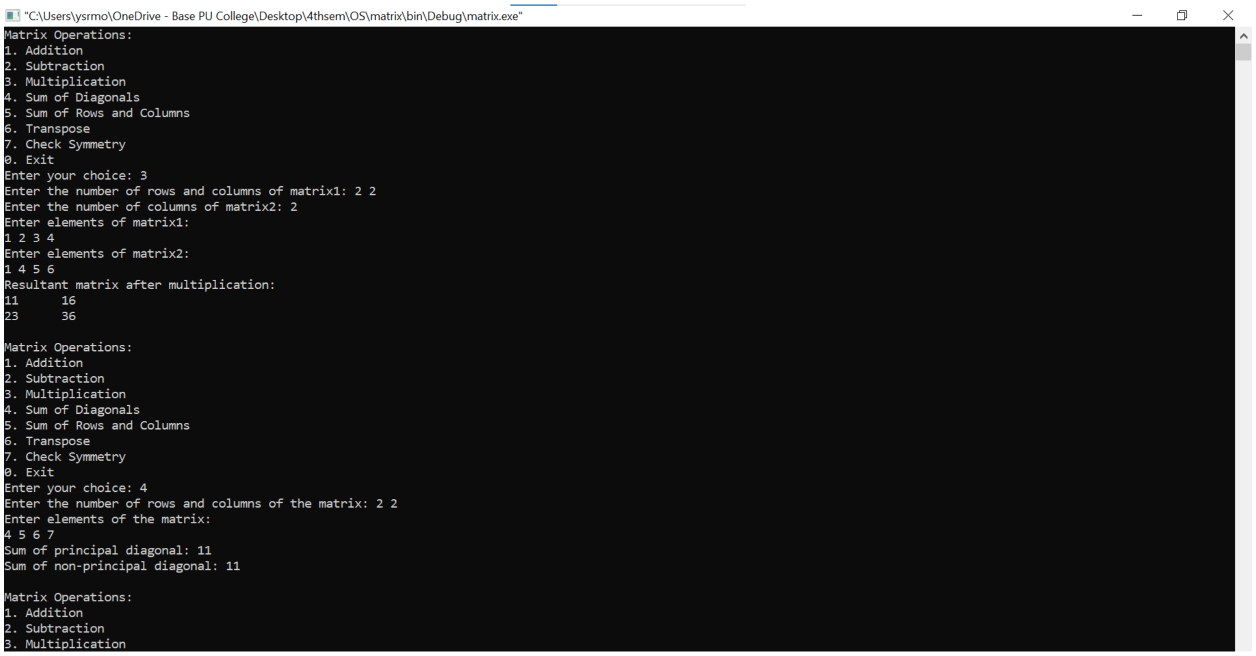
```
            printf("Enter elements of the matrix:\n");
            for (int i = 0; i < rows1; i++) {
                for (int j = 0; j < cols1; j++) {
                    scanf("%d", &matrix1[i][j]);
                }
            }

            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;
}
```

Output:



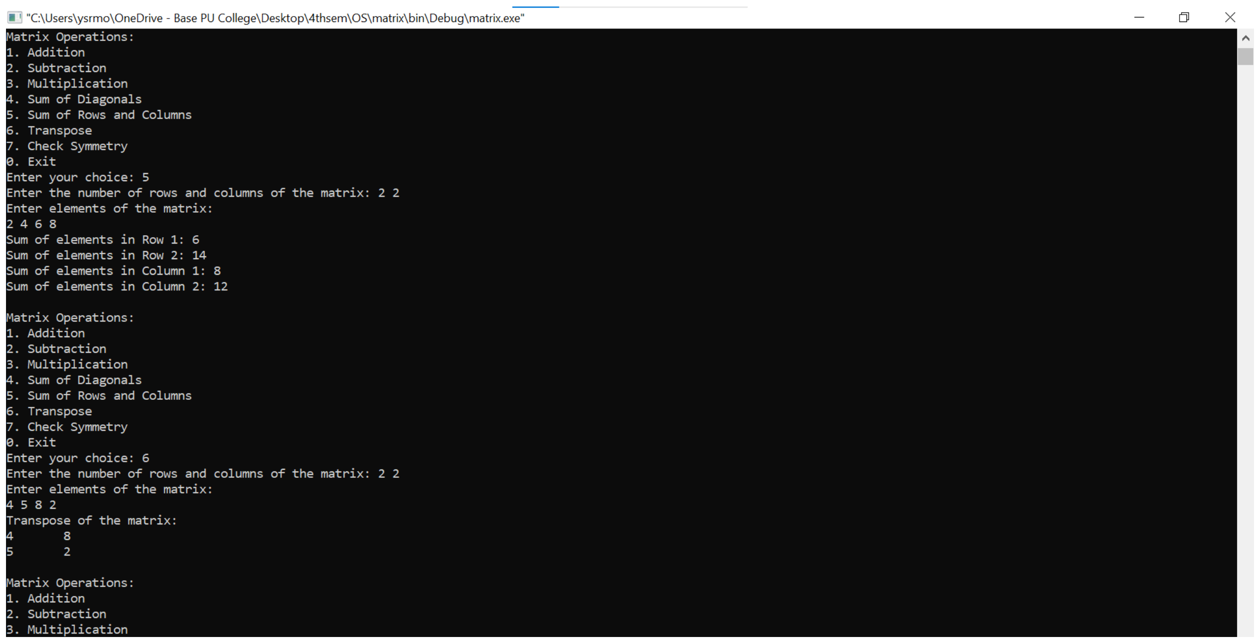
"C:\Users\ysrmo\OneDrive - Base PU College\Desktop\4thsem\OS\matrix\bin\Debug\matrix.exe"

```
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:
6 7 8 9
Enter elements of matrix2:
1 2 3 4
Resultant matrix after subtraction:
5       5
5       5

Matrix Operations:
1. Addition
2. Subtraction
```



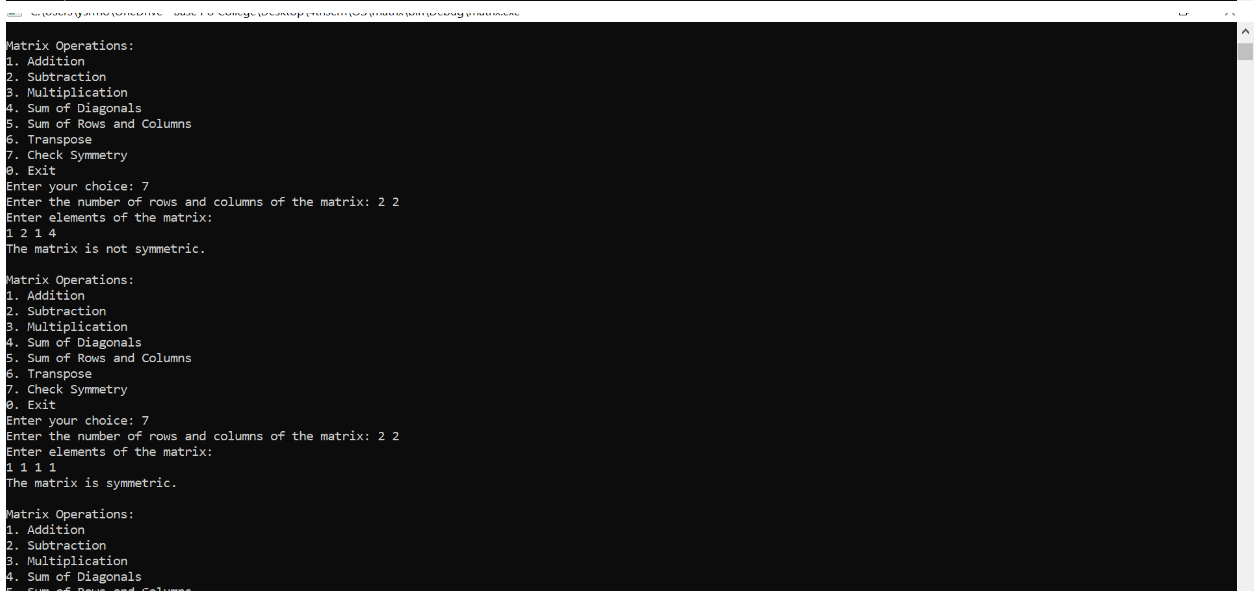
"C:\Users\ysrmo\OneDrive - Base PU College\Desktop\4thsem\OS\matrix\bin\Debug\matrix.exe"

```
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: 3
Enter the number of rows and columns of matrix1: 2 2
Enter the number of columns of matrix2: 2
Enter elements of matrix1:
1 2 3 4
Enter elements of matrix2:
1 4 5 6
Resultant matrix after multiplication:
11      16
23      36

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:
4 5 6 7
Sum of principal diagonal: 11
Sum of non-principal diagonal: 11

Matrix Operations:
1. Addition
2. Subtraction
3. Multiplication
```



"C:\Users\ysrmo\OneDrive - Base PU College\Desktop\4thsem\OS\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: 5
Enter the number of rows and columns of the matrix: 2 2
Enter elements of the matrix:
2 4 6 8
Sum of elements in Row 1: 6
Sum of elements in Row 2: 14
Sum of elements in Column 1: 8
Sum of elements in Column 2: 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: 6
Enter the number of rows and columns of the matrix: 2 2
Enter elements of the matrix:
4 5 8 2
Transpose of the matrix:
4       8
5       2

Matrix Operations:
1. Addition
2. Subtraction
3. Multiplication
```



```
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:
1 2 1 4
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: 7
Enter the number of rows and columns of the matrix: 2 2
Enter elements of the matrix:
1 1 1 1
The matrix is symmetric.

Matrix Operations:
1. Addition
2. Subtraction
3. Multiplication
4. Sum of Diagonals
```