Week - 1
(08 June 2023)
Experiment - 1

Question:

Write a C program to perform the following operations on matrices:
Addition, Subtraction, Multiplication, Row Sum, Column Sum, Transpose, Principal Diagonal Sum, Non-principal Diagonal Sum, Check whether the matrices are symmetrical or not.

Program:

```
#include <stdio.h>

#define MAX_SIZE 10

void matrixAddition(int mat1[][MAX_SIZE], int mat2[][MAX_SIZE], int result[][MAX_SIZE], int rows, int cols) {
    int i, j;
    for (i = 0; i < rows; i++) {
        for (j = 0; j < cols; j++) {
            result[i][j] = mat1[i][j] + mat2[i][j];
        }
    }
}

void matrixSubtraction(int mat1[][MAX_SIZE], int mat2[][MAX_SIZE], int result[][MAX_SIZE], int rows, int cols) {
    int i, j;
    for (i = 0; i < rows; i++) {
        for (j = 0; j < cols; j++) {
            result[i][j] = mat1[i][j] - mat2[i][j];
        }
    }
}

void matrixMultiplication(int mat1[][MAX_SIZE], int mat2[][MAX_SIZE], int result[][MAX_SIZE], int rows1, int cols1, int cols2) {
    int i, j, k;
    for (i = 0; i < rows1; i++) {
        for (j = 0; j < cols2; j++) {
            result[i][j] = 0;
            for (k = 0; k < cols1; k++) {
                result[i][j] += mat1[i][k] * mat2[k][j];
            }
        }
    }
}

int isSymmetric(int mat[][MAX_SIZE], int rows, int cols) {
    int i, j;
    for (i = 0; i < rows; i++) {
        for (j = 0; j < cols; j++) {
            if (mat[i][j] != mat[j][i]) {
                return 0; // Not symmetric
            }
```

```
        }
    }
    return 1; // Symmetric
}

int findPrincipalDiagonalSum(int mat[][MAX_SIZE], int rows, int cols) {
    int i, sum = 0;
    for (i = 0; i < rows && i < cols; i++) {
        sum += mat[i][i];
    }
    return sum;
}

int findNonPrincipalDiagonalSum(int mat[][MAX_SIZE], int rows, int cols) {
    int i, j, sum = 0;
    for (i = 0, j = cols - 1; i < rows && j >= 0; i++, j--) {
        sum += mat[i][j];
    }
    return sum;
}

void findRowSums(int mat[][MAX_SIZE], int rows, int cols, int rowSums[]) {
    int i, j;
    for (i = 0; i < rows; i++) {
        rowSums[i] = 0;
        for (j = 0; j < cols; j++) {
            rowSums[i] += mat[i][j];
        }
    }
}

void findColumnSums(int mat[][MAX_SIZE], int rows, int cols, int colSums[]) {
    int i, j;
    for (j = 0; j < cols; j++) {
        colSums[j] = 0;
        for (i = 0; i < rows; i++) {
            colSums[j] += mat[i][j];
        }
    }
}

void findMatrixTranspose(int mat[][MAX_SIZE], int rows, int cols, int transpose[][MAX_SIZE]) {
    int i, j;
    for (i = 0; i < rows; i++) {
        for (j = 0; j < cols; j++) {
            transpose[j][i] = mat[i][j];
        }
    }
}

void displayMatrix(int mat[][MAX_SIZE], int rows, int cols) {
    int i, j;
    for (i = 0; i < rows; i++) {
        for (j = 0; j < cols; j++) {
            printf("%d ", mat[i][j]);
```

```
        }
        printf("\n");
    }
    printf("\n");
}

int main() {
    int mat1[MAX_SIZE][MAX_SIZE], mat2[MAX_SIZE][MAX_SIZE], result[MAX_SIZE][MAX_SIZE];
    int rows1, cols1, rows2, cols2;

    printf("Enter the number of rows and columns for the first matrix: ");
    scanf("%d %d", &rows1, &cols1);

    printf("Enter the elements of the first matrix:\n");
    int i, j;
    for (i = 0; i < rows1; i++) {
        for (j = 0; j < cols1; j++) {
            scanf("%d", &mat1[i][j]);
        }
    }

    printf("Enter the number of rows and columns for the second matrix: ");
    scanf("%d %d", &rows2, &cols2);

    printf("Enter the elements of the second matrix:\n");
    for (i = 0; i < rows2; i++) {
        for (j = 0; j < cols2; j++) {
            scanf("%d", &mat2[i][j]);
        }
    }

    // Perform matrix addition
    if (rows1 == rows2 && cols1 == cols2) {
        matrixAddition(mat1, mat2, result, rows1, cols1);
        printf("\nResult of matrix addition:\n");
        displayMatrix(result, rows1, cols1);
    } else {
        printf("\nMatrix addition is not possible. The matrices must have the same dimensions.\n");
    }

 // Perform matrix Subtraction
    if (rows1 == rows2 && cols1 == cols2) {
        matrixSubtraction(mat1, mat2, result, rows1, cols1);
        printf("\nResult of matrix subtraction:\n");
        displayMatrix(result, rows1, cols1);
    } else {
        printf("\nMatrix subtraction is not possible. The matrices must have the same dimensions.\n");
    }

    // Perform matrix multiplication
    if (cols1 == rows2) {
        matrixMultiplication(mat1, mat2, result, rows1, cols1, cols2);
        printf("\nResult of matrix multiplication:\n");
        displayMatrix(result, rows1, cols2);
    } else {
```

```
        printf("\nMatrix multiplication is not possible. The number of columns in the first matrix must be equal to the number of rows in the second matrix.\n");
    }

    // Check if matrices are symmetric
    printf("Checking if matrices are symmetric...\n");
    int isMat1Symmetric = isSymmetric(mat1, rows1, cols1);
    int isMat2Symmetric = isSymmetric(mat2, rows2, cols2);

    if (isMat1Symmetric)
        printf("\nMatrix 1 is symmetric.\n");
    else
        printf("\nMatrix 1 is not symmetric.\n");

    if (isMat2Symmetric)
        printf("\nMatrix 2 is symmetric.\n");
    else
        printf("\nMatrix 2 is not symmetric.\n");

    // Find sum of principal and non-principal diagonals of matrix 1
    int principalDiagonalSum1 = findPrincipalDiagonalSum(mat1, rows1, cols1);
    int nonPrincipalDiagonalSum1 = findNonPrincipalDiagonalSum(mat1, rows1, cols1);

    printf("\nSum of principal diagonal of matrix 1: %d\n", principalDiagonalSum1);
    printf("\nSum of non-principal diagonal of matrix 1: %d\n", nonPrincipalDiagonalSum1);

    // Find sum of principal and non-principal diagonals of matrix 2
    int principalDiagonalSum2 = findPrincipalDiagonalSum(mat2, rows2, cols2);
    int nonPrincipalDiagonalSum2 = findNonPrincipalDiagonalSum(mat2, rows2, cols2);

    printf("\nSum of principal diagonal of matrix 2: %d\n", principalDiagonalSum2);
    printf("\nSum of non-principal diagonal of matrix 2: %d\n", nonPrincipalDiagonalSum2);

    // Find sum of every row and every column of matrix 1
    int rowSums[MAX_SIZE], colSums[MAX_SIZE];

    findRowSums(mat1, rows1, cols1, rowSums);
    printf("\nSum of every row of matrix 1:\n");
    for (i = 0; i < rows1; i++) {
        printf("Row %d: %d\n", i + 1, rowSums[i]);
    }

    findColumnSums(mat1, rows1, cols1, colSums);
    printf("\nSum of every column of matrix 1:\n");
    for (j = 0; j < cols1; j++) {
        printf("Column %d: %d\n", j + 1, colSums[j]);
    }

    // Find sum of every row and every column of matrix 2
    findRowSums(mat2, rows2, cols2, rowSums);
    printf("\nSum of every row of matrix 2:\n");
    for (i = 0; i < rows2; i++) {
        printf("Row %d: %d\n", i + 1, rowSums[i]);
    }
```

```
    findColumnSums(mat2, rows2, cols2, colSums);
    printf("\nSum of every column of matrix 2:\n");
    for (j = 0; j < cols2; j++) {
        printf("Column %d: %d\n", j + 1, colSums[j]);
    }

    // transpose of matrix 1
    int transpose1[MAX_SIZE][MAX_SIZE];
    findMatrixTranspose(mat1, rows1, cols1, transpose1);
    printf("\nTranspose of matrix 1:\n");
    displayMatrix(transpose1, cols1, rows1);


    int transpose2[MAX_SIZE][MAX_SIZE];
    findMatrixTranspose(mat2, rows2, cols2, transpose2);
    printf("\nTranspose of matrix 2:\n");
    displayMatrix(transpose2, cols2, rows2);

    return 0;
}
```

Output:

```
Enter the number of rows and columns for the first matrix: 2 2
Enter the elements of the first matrix:
1 2
3 4
Enter the number of rows and columns for the second matrix: 2 2
Enter the elements of the second matrix:
5 6
7 8

Result of matrix addition:
6 8
10 12


Result of matrix subtraction:
-4 -4
-4 -4


Result of matrix multiplication:
19 22
43 50

Checking if matrices are symmetric...

Matrix 1 is not symmetric.

Matrix 2 is not symmetric.
```

```
Sum of principal diagonal of matrix 1: 5

Sum of non-principal diagonal of matrix 1: 5

Sum of principal diagonal of matrix 2: 13

Sum of non-principal diagonal of matrix 2: 13

Sum of every row of matrix 1:
Row 1: 3
Row 2: 7

Sum of every column of matrix 1:
Column 1: 4
Column 2: 6

Sum of every row of matrix 2:
Row 1: 11
Row 2: 15

Sum of every column of matrix 2:
Column 1: 12
Column 2: 14

Transpose of matrix 1:
1 3
2 4


Transpose of matrix 2:
5 7
6 8
```

Observation Book Pictures:

PAGE NO :
DATE : 08/06/2023

Experiment - 1

Write a C program to perform the following operations on matrices:
Addition, Multiplication, Subtraction, Row sum, Column Sum, Transpose, principal diagnol sum, non principal diagnol sum, Check whether the matrices are symmetric or not.

Program:

```
#include<stdio.h>
#define MAX_SIZE 10

void matrixAddition(int mat1[][MAX_SIZE],
int mat2[][MAX_SIZE], int result[][MAX_SIZE],
int rows, int cols)
{  int i, j;
   for(i=0; i<rows, i++) {
      for(j=0; j<cols; j++)
         { result[i][j] = mat1[i][j] + mat2[i][j];
         }
      }
}

void matrixSubtraction(int mat1[][MAX_SIZE],
int mat2[][MAX_SIZE], int result[][MAX_SIZE],
int rows, int cols)
{ int i, j;
   for(i=0; i<rows; i++) {
      for(j=0; j<cols; j++)
         { result[i][j] = mat1[i][j] - mat2[i][j];
         }
      }
}
```

PAGE NO :
DATE :

```
void matrixMultiplication(int mat1[][MAX-SIZE],
int mat2[][MAX-SIZE], int result[][MAX-SIZE],
int rows1, int cols1, int cols2)
{ int i, j, k;
    for (i=0; i<rows1; i++) {
      for (j=0; j<cols2; j++) {
        result[i][j] = 0;
          for (k=0; k<cols1; k++) {
            result[i][j] += mat1[i][j] * mat2[i][j];
          }
      }
    }
}

int isSymmetric(int mat[][MAX-SIZE], int rows,
int cols)
{ int i, j;
   for (i=0; i<rows; i++) {
     for (j=0; j<cols; j++) {
       if mat[i][j] != mat[j][i])
          { return 0; // not symmetric
          }
     }
}
return 1; // symmetric
}

int findPrincipalDiagonalSum(int[][MAX-SIZE],
int rows, int cols)
{ int i, sum = 0;
    for (i=0; i<rows && i<cols; i++)
      { sum += mat[i][i];
      } return sum;
}
```

PAGE NO :
DATE :

```
int findNonPrincipalDiagnolSum (int mat[][MAX-SIZE],
int rows, int cols)
{  int i, j, sum = 0;
    for (i=0; j=cols-1; i<rows && j>=0;
         i++, j--)
    {  sum += mat[i][j];
    }  return sum;
}

void findRowSums (int mat[][MAX-SIZE], int rows,
int cols; int rowSums[])
{  int i, j;
    for (i=0; i<rows; i++) {
        rowSums[i] = 0;
        for (j=0; j<cols; j++) {
        rowSums[i] += mat[i][j];
        }
    }
}

void findColumnSums(int mat[][MAX-SIZE], int rows,
int cols, int colSums[])
{  int i, j;
    for (j=0; j<cols; j++) {
        colSums[j] = 0;
        for (i=0; i<rows; i++) {
            colSums[j] += mat[i][j];
        }
    }
}
```

```
void findMatrixTranspose(int mat[][MAX_SIZE],
int rows, int cols, int transpose[][MAX_SIZE])
{ int i, j;
   for(i=0; i<rows; i++) {
      for(j=0; j<cols; j++) {
         transpose[j][i] = mat[i][j];
      }
   }
}

void displayMatrix(int mat[][MAX_SIZE], int rows,
int cols)
{   for(i=0; i<rows; i++) {
      for(j=0; j<cols; j++) {
         printf("%d", mat[i][j]);
      }
      printf("\n");
   }
   printf("\n");
}

int main()
{ int mat1[MAX_SIZE][MAX_SIZE],
      mat2[MAX_SIZE][MAX_SIZE],
      result[MAX_SIZE][MAX_SIZE];
  int rows1, cols1, rows2, cols2;

  printf("Enter the no of rows and columns
         for the first matrix:");
  scanf("%d %d", &rows1, &cols1);
```

→

```
printf(" Enter the elements of the first matrix :\n");
int i, j;
for (i = 0; i < rows1 ; i++){
    for (j = 0; j < cols1 ;j++) {
        scanf("%d", &mat1[i][j]);
    }
}

printf("Enter the no of rows and columns for
        the second matrix : ");
scanf("%d %d", &rows2, &cols2);

printf(" Enter the elements of the second matrix :\n");
for (i=0; i < rows2 ; i++){
    for (j=0;j<cols2 ; j++ ) {
        scanf("%d", &mat2[i][j]);
    }
}

// matrix addition
if (rows1 == rows2 && cols1 == cols2)
{ matrixAddition(mat1, mat2, result, rows1,
                                    cols1 );
  printf("\n Result of matrix addition :\n");
  displayMatrix(result, rows1, cols1);
}
else
{ printf("\n Matrix addition is not possible. The
          matrices must have the same dimensions.
          \n");
}
```

```
// matrix Subtraction
if (rows1 = rows2 && cols1 == cols2)
{ matrixSubtraction(mat1, mat2, result,
                    rows1, cols1);
  printf("\nResult of matrix Subtraction:\n");
  displayMatrix(result, rows1, cols1);
}
else
{ printf("\nMatrix subtraction is not possible.
          The matrices must have the same
          dimensions.\n");
}

// matrix multiplication
if (cols1 == rows2);
{ matrixMultiplication(mat1, mat2, result,
                       rows1, cols1, cols2);
  printf("\n Result of Matrix Multiplication:\n");
  displayMatrix(result, rows1, cols2);
}
else
{ printf("\n Matrix Multiplication is not possible");

// check if matrices are symmetric
printf("Checking if matrices are symmetric.\n");
int isMat1Symmetric = isSymmetric(mat1, rows1, cols1);
int isMat2Symmetric = isSymmetric(mat2, rows2, cols2);

if (isMat1Symmetric)
  printf("\n Matrix 1 is symmetric\n");
else
  printf("\n Matrix 1 is not symmetric\n");
```

```
if (isMat2Symmetric)
    printf ("\n Matrix 2 is symmetric \n");
else
    printf ("\n Matrix 2 is not symmetric \n");

// Sum of principal and non principal diagnols of
// matrix 1 and matrix 2

int principalDiagnolSum1= findPrincipalDiagnolSum(
                          mat1, rows1, cols1);
int nonprincipalDiagnolSum1= findNonPrincipalDiagnolSum(
                          mat1, rows1, cols1);

printf(" Sum of principal Diagnol Sum of matrix 1:
        %d\n", principalDiagnolSum1);
printf (" Sum of Nonprincipal Diagnol Sum of matrix 1:
        %d\n", nonprincipalDiagnolSum1);

int principalDiagnolSum2 = findPrincipalDiagnolSum(
                          mat2, rows2, cols2);
int nonprincipalDiagnolSum2 = findNonPrincipalDiagnolSum(
                          mat2, rows2, cols2);

printf("Sum of principal Diagnol Sum of matrix 2:
        %d\n", principalDiagnolSum2);
printf (" Sum of Non principal Diagnol Sum of matrix 2:
        %d\n", nonprincipalDiagnolSum2);

// Sum of every row and column of matrix 1
int rowSums[MAX-SIZE], colSums[MAX-SIZE];
findRowSums(mat1, rows1, cols1, rowSums);
printf ("\n Sum of every row of matrix 1:\n");
```

→

```
for (i=0; i<rows1; i++)
{ printf("Row %d: %d\n", i+1, rowSums[i]);
}

findColumnSums(mat1, rows1, cols1, colSums);
printf("\n Sum of every column of matrix 1:\n");
for (j=0; j<cols1; j++)
{ printf("Column %d: %d\n", j+1, colSums[j]);

// Sum of every row and column of matrix 2
findRowSums(mat2, rows2, rowSums);
printf("\n Sum of every row of matrix 2:\n");
for (i=0; i<rows2; i++)
{ printf("Row %d: %d\n", i+1, rowSums[i]);
}

findColumnSums(mat2, rows2, cols2, colSums);
printf("\n Sum of every column of matrix 2:\n");
for (j=0; j<cols2; j++)
{ printf("Column %d: %d\n", j+1, colSums[j]);
}

// transpose of matrix 1 and 2.
int transpose1[MAX_SIZE][MAX_SIZE];
findMatrixTranspose(mat1, rows1, cols1, transpose1);
printf("\n Transpose of matrix 1:\n");
displayMatrix(transpose1, cols1, rows1);

int transpose2[MAX_SIZE][MAX_SIZE];
findMatrixTranspose(mat2, rows2, cols2, transpose2);
printf("\n Transpose of matrix 2:\n");
displayMatrix(transpose2, cols2, rows2);
return 0;
}
```

DATE :

Output:

Enter the no. of rows and columns for the first matrix : 2 2

Enter the elements of first matrix:

1 2

3 4

Enter the no. of rows and columns for the second matrix : 2 2

5 6

7 8

Result of Matrix Addition:

6 8

10 12

Result of Matrix Subtraction:

-4 -4

-4 -4

Result of Matrix Multiplication:

19 22

43 50

Checking if matrices are Symmetric..

Matrix 1 is not symmetric.

Matrix 2 is not symmetric.

Sum of principal Diagnol of Matrix 1 : 5

Sum of Non-principal Diagnol of Matrix 1 : 5

Sum of principal Diagnol of Matrix 2 : 13

Sum of Non-principal Diagnol of Matrix 2 : 13

→

Sum of every row of Matrix 1 :
Row 1 : 3
Row 2 : 7

Sum of every Column of Matrix 1 :
Column 1 : 4
Column 2 : 6

Sum of every row of Matrix 2 :
Row 1 : 11
Row 2 : 15

Sum of every Column of Matrix 2 :
Column 1 : 12
Column 2 : 14

Transpose of matrix 1 :
1 3
2 4

Transpose of matrix 2 :
5 7
6 8

22/6/23