**Data Structure**

**Lab-5**

**Submitted by: Submitted to:**

**Aakash Bhatt Pankaj Sir**

**500124633**

**(2023-2024)**

A. WAP to implement the following scenarios. Take all the input from user, nothing should be imagined or hard coded.

1. Transpose of a matrix

#include <stdio.h>

int main()

{

    int arr[2][3], i, j;

*// Input values for the matrix*

    printf("Enter values for a 2x3 matrix:\n");

    for (i = 0; i < 2; i++)

  {

      for (j = 0; j < 3; j++)

        {

        scanf("%d", &arr[i][j]);

        }

    }

*// Display the original matrix*

    printf("The matrix is:\n");

    for (i = 0; i < 2; i++)

    {

        for (j = 0; j < 3; j++)

        {

            printf("%d\t", arr[i][j]);

        }

        printf("\n");

    }

*// Display the transpose of the matrix*

    printf("The transpose of the matrix is:\n");

  for (i = 0; i < 3; i++)

  {

        for (j = 0; j < 2; j++)

        {

            printf("%d\t", arr[j][i]);

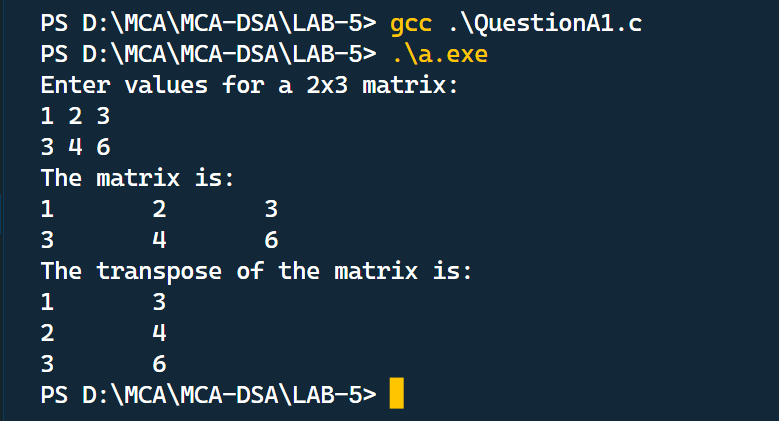
        }

        printf("\n");

    }

return 0;

}



2.Check if a matrix is Syymetrical or not

#include <stdio.h>

#include <stdbool.h>

void main()

{

    int a[3][3], i, j;

    bool isSymmetric = true;

*// Input values for the matrix*

    printf("Enter the elements of the matrix:\n");

    for (i = 0; i < 3; i++)

    {

        for (j = 0; j < 3; j++)

        {

            scanf("%d", &a[i][j]);

        }

    }

*// Display the elements of the matrix*

    printf("The elements of the matrix are:\n");

    for (i = 0; i < 3; i++)

    {

        for (j = 0; j < 3; j++)

        {

            printf("%d\t", a[i][j]);

        }

        printf("\n");

    }

*// Display the transpose of the matrix*

    printf("Transpose of the matrix is:\n");

    for (i = 0; i < 3; i++)

    {

        for (j = 0; j < 3; j++)

        {

            printf("%d\t", a[j][i]);

        }

        printf("\n");

    }

*// Check if the matrix is symmetric*

    for (i = 0; i < 3; i++)

    {

        for (j = 0; j < 3; j++)

        {

            if (a[i][j] != a[j][i])

            {

                isSymmetric = false;

                break; *// Exit the loop as soon as a non-symmetric element is found*

            }

        }

    }

*// Check and display if the matrix is symmetric or not*

    if (isSymmetric)

    {

        printf("Matrix is symmetric.\n");

    }

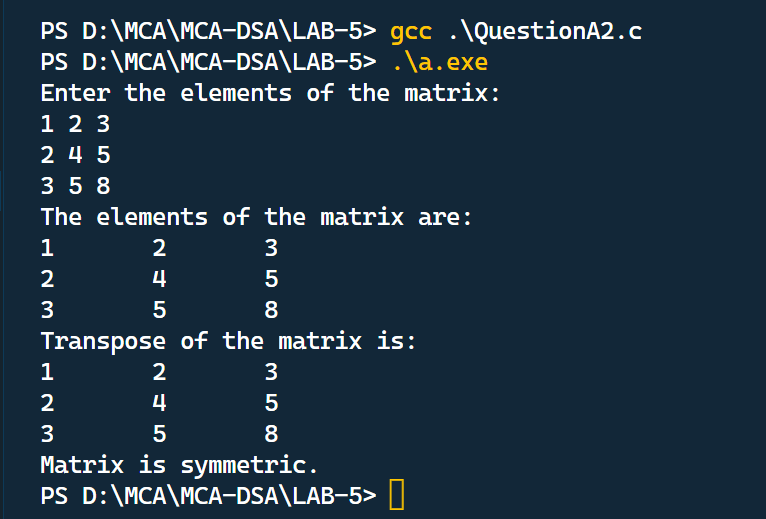
    else

    {

        printf("Matrix isn't symmetric.\n");

    }

}



3.Check the inverse of a matrix

#include <stdio.h>

*// Function to print a 3x3 matrix*

void printMatrix(double A[3][3])

{

    for (int i = 0; i < 3; i++)

    {

        for (int j = 0; j < 3; j++)

        {

            printf("%.2f\t", A[i][j]);

        }

        printf("\n");

    }

}

*// Global variable to store the determinant*

double det = 0;

*// Function to calculate the inverse of a 3x3 matrix*

void inverseMatrix(double A[3][3], double A\_inv[3][3])

{

*// Calculate the determinant of A using the formula for a 3x3 matrix*

    for (int i = 0; i < 3; i++)

    {

        det += (A[0][i] \* (A[1][(i + 1) % 3] \* A[2][(i + 2) % 3] - A[1][(i + 2) % 3] \* A[2][(i + 1) % 3]));

    }

*// Check if the matrix is singular (determinant is zero)*

    if (det == 0)

    {

        printf("Matrix is singular. Inverse does not exist.\n");

        return;

    }

*// Calculate the inverse of A using the adjugate and determinant*

    for (int i = 0; i < 3; i++)

    {

        for (int j = 0; j < 3; j++)

        {

            A\_inv[i][j] = ((A[(j + 1) % 3][(i + 1) % 3] \* A[(j + 2) % 3][(i + 2) % 3]) -

                           (A[(j + 1) % 3][(i + 2) % 3] \* A[(j + 2) % 3][(i + 1) % 3])) /

                          det;

        }

    }

}

int main()

{

    double A[3][3] = {{2.0, 3.0, 4.0},

                      {1.0, 5.0, 6.0},

                      {7.0, 8.0, 9.0}};

    double A\_inv[3][3];

    printf("Original Matrix A:\n");

    printMatrix(A);

    inverseMatrix(A, A\_inv);

*// Check if the determinant is nonzero (inverse exists)*

    if (det != 0)

    {

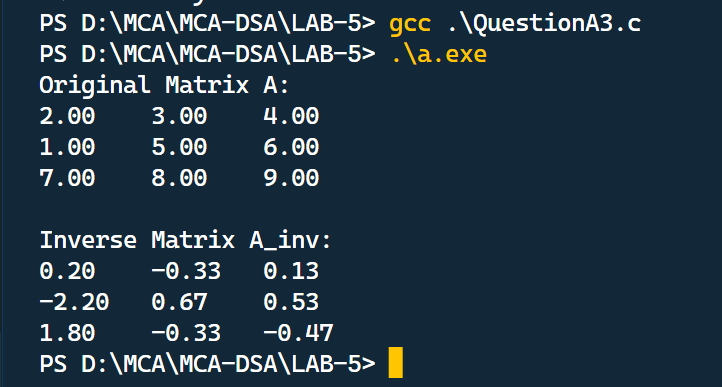
        printf("\nInverse Matrix A\_inv:\n");

        printMatrix(A\_inv);

    }

    return 0;

}



B. WAP to merge two arrays and append them in the following order.

1. Add the first array to the end of another one
2. Add Second Array to the end of the first one
3. Merge the arrays and sort them.

#include <stdio.h>

int main()

{

*// Define and initialize the first array (arr1)*

    int arr1*[]* = {5, 2, 9};

    int size1 = sizeof(arr1) / sizeof(arr1[0]);

*// Define and initialize the second array (arr2)*

    int arr2*[]* = {7, 1, 3};

    int size2 = sizeof(arr2) / sizeof(arr2[0]);

*// Calculate the size of the merged array*

    int mergedSize = size1 + size2;

*// Create an array to store the merged elements*

    int mergedArray[mergedSize];

*// Copy elements from arr1 and arr2 to mergedArray*

    for (int i = 0; i < size1; i++)

    {

        mergedArray[i] = arr1[i];

    }

    for (int i = 0; i < size2; i++)

    {

        mergedArray[size1 + i] = arr2[i];

    }

*// Sort the merged array using bubble sort*

    for (int i = 0; i < mergedSize - 1; i++)

    {

        for (int j = 0; j < mergedSize - i - 1; j++)

        {

            if (mergedArray[j] > mergedArray[j + 1])

            {

*// Swap elements if they are in the wrong order*

                int temp = mergedArray[j];

                mergedArray[j] = mergedArray[j + 1];

                mergedArray[j + 1] = temp;

            }

        }

    }

*// Print the sorted merged array*

    printf("Merged and sorted array:\n");

    for (int i = 0; i < mergedSize; i++)

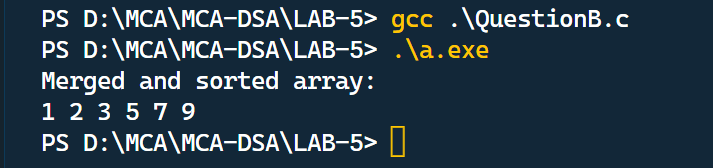
    {

        printf("%d ", mergedArray[i]);

    }

    return 0;

}



C. WAP using pointers to find the smallest number in an array using pointer.

#include <stdio.h>

int main()

{

    int num;

*// Prompt the user to enter the size of the array*

    printf("Enter the size of array: ");

    scanf("%d", &num);

*// Declare an integer array of the given size and a pointer to an integer*

    int arr[num], \*small;

*// Input elements into the array*

    for (int i = 0; i < num; i++)

    {

        scanf("%d", &arr[i]);

    }

*// Initialize the 'small' pointer to point to the first element of the array*

    small = &arr[0];

*// Find the smallest element in the array using pointer arithmetic*

    for (int i = 0; i < num; i++)

    {

        if (\*(arr + i) < \*small)

            \*small = \*(arr + i);

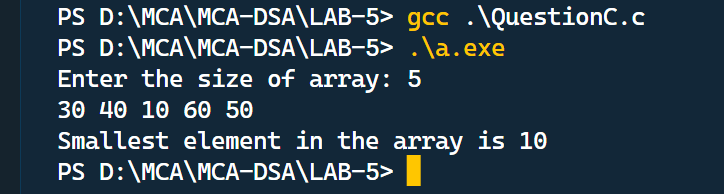
    }

*// Print the smallest element in the array*

    printf("Smallest element in the array is %d", \*small);

    return 0;

}



D. WAP which performs following task.

1. Initialize an integer array of 10 elements in main()
2. Pass the entire array to a function modify()
3. In modify() multiply(you can use division, addition or subtraction) each element of array by 3
4. Return the control to main() and print the new array elements in main().

#include <stdio.h>

*// Function to modify the array elements by multiplying them by 3*

void modify(int arr*[]*, int size)

{

    for (int i = 0; i < size; i++)

    {

        arr[i] \*= 3; *// Multiply each element by 3*

    }

}

int main()

{

    int arr[10]; *// Initialize an integer array of 10 elements*

*// Initialize the array elements in main()*

    printf("Enter 10 integers:\n");

    for (int i = 0; i < 10; i++)

    {

        scanf("%d", &arr[i]);

    }

*// Call the modify function to multiply each element by 3*

    modify(arr, 10);

*// Print the modified array elements in main()*

    printf("Modified array elements:\n");

    for (int i = 0; i < 10; i++)

    {

        printf("%d ", arr[i]);

    }

    return 0;

}

