#include <stdio.h>

#define MAX\_SIZE 10 // Defining the maximum size allowed for matrices

// Function prototypes

void inputMatrix(int rows, int cols, int matrix[MAX\_SIZE][MAX\_SIZE]);

void printMatrix(int rows, int cols, int matrix[MAX\_SIZE][MAX\_SIZE]);

void addMatrices(int rows, int cols, int A[MAX\_SIZE][MAX\_SIZE], int B[MAX\_SIZE][MAX\_SIZE], int result[MAX\_SIZE][MAX\_SIZE]);

void subtractMatrices(int rows, int cols, int A[MAX\_SIZE][MAX\_SIZE], int B[MAX\_SIZE][MAX\_SIZE], int result[MAX\_SIZE][MAX\_SIZE]);

void transposeMatrix(int rows, int cols, int matrix[MAX\_SIZE][MAX\_SIZE], int result[MAX\_SIZE][MAX\_SIZE]);

int main() {

int rowsA, colsA, rowsB, colsB; // Dimensions for matrices A and B

int A[MAX\_SIZE][MAX\_SIZE], B[MAX\_SIZE][MAX\_SIZE]; // Matrices A and B

int result[MAX\_SIZE][MAX\_SIZE]; // Matrix to store results

int choice = 0; // User's choice for menu options

// Prompt the user for input of dimensions for Matrix A

printf("Enter dimensions for Matrix A (rows and columns): ");

scanf("%d %d", &rowsA, &colsA);

// Validate dimensions for Matrix A

while (rowsA < 1 || rowsA > MAX\_SIZE || colsA < 1 || colsA > MAX\_SIZE) {

printf("Invalid dimensions. Try again.\n");

printf("Enter dimensions for Matrix A (rows and columns): ");

scanf("%d %d", &rowsA, &colsA);

}

// Prompt the user for input of elements for Matrix A

printf("Enter elements for Matrix A:\n");

inputMatrix(rowsA, colsA, A);

// // Prompt the user for input of dimensions for Matrix A

printf("Enter dimensions for Matrix B (rows and columns): ");

scanf("%d %d", &rowsB, &colsB);

// Validate dimensions for Matrix B

while (rowsB < 1 || rowsB > MAX\_SIZE || colsB < 1 || colsB > MAX\_SIZE) {

printf("Invalid dimensions. Try again.\n");

printf("Enter dimensions for Matrix B (rows and columns): ");

scanf("%d %d", &rowsB, &colsB);

}

// Prompt the user for input of elements for Matrix B

printf("Enter elements for Matrix B:\n");

inputMatrix(rowsB, colsB, B);

// Menu for the user to choose a matrix operation

while (choice != 5) {

// Display the menu

printf("\nMatrix Operations Menu:\n");

printf("1. Add Matrices\n");

printf("2. Subtract Matrices\n");

printf("3. Transpose Matrix A\n");

printf("4. Transpose Matrix B\n");

printf("5. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

// Perform the matrix operation based on the user's choice

if (choice == 1) {

// Matrix addition

if (rowsA == rowsB && colsA == colsB) {

addMatrices(rowsA, colsA, A, B, result);

printf("Result of Matrix A + Matrix B:\n");

printMatrix(rowsA, colsA, result);

} else {

printf("Matrices dimensions do not match for addition.\n");

}

} else if (choice == 2) {

// Matrix subtraction

if (rowsA == rowsB && colsA == colsB) {

subtractMatrices(rowsA, colsA, A, B, result);

printf("Result of Matrix A - Matrix B:\n");

printMatrix(rowsA, colsA, result);

} else {

printf("Matrices dimensions do not match for subtraction.\n");

}

} else if (choice == 3) {

// Transpose of Matrix A

transposeMatrix(rowsA, colsA, A, result);

printf("Transpose of Matrix A:\n");

printMatrix(colsA, rowsA, result);

} else if (choice == 4) {

// Transpose of Matrix B

transposeMatrix(rowsB, colsB, B, result);

printf("Transpose of Matrix B:\n");

printMatrix(colsB, rowsB, result);

} else if (choice == 5) {

// Exit the program

printf("Exiting...\n");

} else {

// Invalid choice

printf("Invalid choice. Please try again.\n");

}

}

return 0;

}

// Function to input elements of a matrix

void inputMatrix(int rows, int cols, int matrix[MAX\_SIZE][MAX\_SIZE]) {

// Loop through each element of the matrix

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

printf("Enter element [%d][%d]: ", i + 1, j + 1);

scanf("%d", &matrix[i][j]);

}

}

}

// Function to print elements of a matrix

void printMatrix(int rows, int cols, int matrix[MAX\_SIZE][MAX\_SIZE]) {

// Loop through each element and print in matrix form

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

printf("%d ", matrix[i][j]);

}

printf("\n");

}

}

// Function to add Matrix A to Matrix B

void addMatrices(int rows, int cols, int A[MAX\_SIZE][MAX\_SIZE], int B[MAX\_SIZE][MAX\_SIZE], int result[MAX\_SIZE][MAX\_SIZE]) {

// Add corresponding elements from matrices A and B

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

result[i][j] = A[i][j] + B[i][j];

}

}

}

// Function to subtract Matrix B from Matrix A

void subtractMatrices(int rows, int cols, int A[MAX\_SIZE][MAX\_SIZE], int B[MAX\_SIZE][MAX\_SIZE], int result[MAX\_SIZE][MAX\_SIZE]) {

// Subtract corresponding elements of B from A

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

result[i][j] = A[i][j] - B[i][j];

}

}

}

// Function to compute the transpose of a matrix

void transposeMatrix(int rows, int cols, int matrix[MAX\_SIZE][MAX\_SIZE], int result[MAX\_SIZE][MAX\_SIZE]) {

// Transpose the matrix by swapping rows and columns

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

result[j][i] = matrix[i][j];

}

}

}