#include <stdio.h>

#include <stdlib.h>

#define MAX\_SIZE 10

// Function to get a positive integer input

int getPositiveInteger(const char\* prompt) {

int num;

do {

printf(prompt);

if (scanf("%d", &num)!= 1 || num <= 0) {

while (getchar()!= '\n');// Clear the input buffer

printf("Please enter a positive integer.\n");

} else {

break;

}

} while (1);

return num;

}

// Function to create a 2D matrix

int\*\* createMatrix(int rows, int cols) {

int\*\* matrix = (int\*\*) malloc(rows \* sizeof(int\*));

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

matrix[i] = (int\*) malloc(cols \* sizeof(int));

}

return matrix;

}

// Function to free the memory of a 2D matrix

void freeMatrix(int\*\* matrix, int rows) {

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

free(matrix[i]);

}

free(matrix);

}

// Function to input elements of a matrix

void inputMatrix(int\*\* matrix, int rows, int cols) {

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

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

int inputStatus;

do {

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

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

if (inputStatus!= 1) {

while (getchar()!= '\n');// Clear the input buffer

printf("Input error. Please re - enter an integer.\n");

}

} while (inputStatus!= 1);

}

}

}

// Function to print a matrix

void printMatrix(int\*\* matrix, int rows, int cols) {

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

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

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

}

printf("\n");

}

}

// Function to add two matrices

void addMatrices(int\*\* matrixA, int\*\* matrixB, int\*\* result, int rows, int cols) {

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

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

result[i][j] = matrixA[i][j] + matrixB[i][j];

}

}

}

// Function to subtract two matrices

void subtractMatrices(int\*\* matrixA, int\*\* matrixB, int\*\* result, int rows, int cols) {

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

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

result[i][j] = matrixA[i][j] - matrixB[i][j];

}

}

}

// Function to transpose a matrix

void transposeMatrix(int\*\* matrix, int\*\* result, int rows, int cols) {

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

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

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

}

}

}

int main() {

int\*\* matrixA, \*\*matrixB, \*\*result;

int rowsA, colsA, rowsB, colsB;

int choice;

// The user enters the dimensions of matrix A

rowsA = getPositiveInteger("Enter rows for Matrix A: ");

colsA = getPositiveInteger("Enter columns for Matrix A: ");

if (rowsA > MAX\_SIZE || colsA > MAX\_SIZE) {

printf("Matrix dimensions exceed the maximum size of %dx%d.\n", MAX\_SIZE, MAX\_SIZE);

return 1;

}

matrixA = createMatrix(rowsA, colsA);

// The user enters the elements of matrix A

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

inputMatrix(matrixA, rowsA, colsA);

// The user enters the dimensions of matrix B

rowsB = getPositiveInteger("Enter rows for Matrix A: ");

colsB = getPositiveInteger("Enter columns for Matrix B: ");

if (rowsB > MAX\_SIZE || colsB > MAX\_SIZE) {

printf("Matrix dimensions exceed the maximum size of %dx%d.\n", MAX\_SIZE, MAX\_SIZE);

return 1;

}

matrixB = createMatrix(rowsB, colsB);

// The user enters the elements of matrix B

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

inputMatrix(matrixB, rowsB, colsB);

result = createMatrix(rowsA > rowsB? rowsA : rowsB, colsA > colsB? colsA : colsB);

do {

// Display 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);

switch (choice) {

case 1: // Matrix addition

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

addMatrices(matrixA, matrixB, result, rowsA, colsA);

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

printMatrix(result, rowsA, colsA);

} else {

printf("Matrices dimensions must be the same for addition.\n");

}

break;

case 2: // Matrix subtraction

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

subtractMatrices(matrixA, matrixB, result, rowsA, colsA);

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

printMatrix(result, rowsA, colsA);

} else {

printf("Matrices dimensions must be the same for subtraction.\n");

}

break;

case 3:

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

transposeMatrix(matrixA, result, rowsA, colsA);

printMatrix(result, colsA, rowsA);

break;

case 4:

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

transposeMatrix(matrixB, result, rowsB, colsB);

printMatrix(result, colsB, rowsB);

break;

case 5:

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

break;

default:

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

}

} while (choice!= 5);

// Free matrix memory

freeMatrix(matrixA, rowsA);

freeMatrix(matrixB, rowsB);

freeMatrix(result, rowsA > rowsB? rowsA : rowsB);

return 0;

}