#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include<stdlib.h>

#include<ctype.h>

#include<math.h>

#include<string.h>

void matrixAddSub(int arrayone[10][10], int arraytwo[10][10], int rows, int column, int mul);

void matrixScalarMultiply(int array[10][10], int scalar, int rows, int column);

void matrixMultiply(int arrayone[10][10], int arraytwo[10][10], int rowsA, int columnA, int columnB);

int main(void)

{

int operation;

char again = 'Y';

int scalar = 0;

int add = 1;

int sub = -1;

int i,j,k;

while (again == 'Y')

{

printf("\nOperation Menu\n");

printf("\t1. to Add\n");

printf("\t2. to Subtract\n");

printf("\t3. to Scalar Multiply\n");

printf("\t4. to Multiply two matrices\n");

printf("Enter yout choice: ");

scanf(" %d", &operation);

switch (operation)

{

case 1:

int i, j, k;

int matrixA[10][10];

int matrixB[10][10];

int rowA, columnA;

int rowB, columnB;

printf("\nEnter the #rows and #cols for matrix A: ");

scanf("%d%d", &rowA, &columnA);

printf("Enter the #rows and #cols for matrix B: ");

scanf("%d%d", &rowB, &columnB);

while ((rowA != rowB) && (columnA != columnB))

{

printf("\nMatrices must be the same size\n");

printf("\nEnter the #rows and #cols for matrix A: ");

scanf("%d%d", &rowA, &columnA);

printf("Enter the #rows and #cols for matrix B: ");

scanf("%d%d", &rowB, &columnB);

}

printf("\n\tEnter elements of Matrix A a %d x %d matrix.\n", rowA, columnA);

for (i = 0; i < rowA; i++)

{

printf("\t%d entries for row %d: ", columnA, i + 1);

for (j = 0; j < columnA; j++)

{

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

}

}

printf("\n\t\tMatrix A\n\n");

for (i = 0; i < rowA; i++)

{

for (j = 0; j < columnA; j++)

{

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

}

printf("\n");

}

printf("\n\tEnter elements of Matrix B a %d x %d matrix.\n", rowB, columnB);

for (i = 0; i < rowB; i++)

{

printf("\t%d entries for row %d: ", columnB, i + 1);

for (j = 0; j < columnB; j++)

{

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

}

}

printf("\n\t\tMatrix B\n\n");

for (i = 0; i < rowB; i++)

{

for (j = 0; j < columnB; j++)

{

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

}

printf("\n");

}

printf("\nThe Sum of matrixA + matrixB is : \n");

matrixAddSub(matrixA, matrixB, rowA, columnA, add);

break;

case 2:

int matrixC[10][10];

int matrixD[10][10];

int rowC, columnC;

int rowD, columnD;

int a,b;

printf("\nEnter the #rows and #cols for matrix C: ");

scanf("%d%d", &rowC, &columnC);

printf("Enter the #rows and #cols for matrix D: ");

scanf("%d%d", &rowD, &columnD);

while ((rowC != rowD) && (columnC != columnD))

{

printf("\nMatrices must be the same size\n");

printf("\nEnter the #rows and #cols for matrix C: ");

scanf("%d%d", &rowC, &columnC);

printf("Enter the #rows and #cols for matrix D: ");

scanf("%d%d", &rowD, &columnD);

}

printf("\n\tEnter elements of Matrix C a %d x %d matrix.\n", rowC, columnC);

for (a = 0; a < rowC; a++)

{

printf("\t%d entries for row %d: ", columnC, a + 1);

for (b = 0; b < columnC; b++)

{

scanf("%d", &matrixC[a][b]);

}

}

printf("\n\t\tMatrix C\n\n");

for (a = 0; a < rowC; a++)

{

for (b = 0; b < columnC; b++)

{

printf("\t%d", matrixC[a][b]);

}

printf("\n");

}

printf("\n\tEnter elements of Matrix D a %d x %d matrix.\n", rowD, columnD);

for (a = 0; a < rowD; a++)

{

printf("\t%d entries for row %d: ", columnD, a + 1);

for (b = 0; b < columnD; b++)

{

scanf("%d", &matrixD[a][b]);

}

}

printf("\n\t\tMatrix D\n\n");

for (a = 0; a < rowD; a++)

{

for (b = 0; b < columnD; b++)

{

printf("\t%d", matrixD[a][b]);

}

printf("\n");

}

printf("\nThe difference between matrixC- matrix is : \n");

matrixAddSub(matrixC, matrixD, rowC, columnC, sub);

break;

case 3:

int matrixE[10][10];

int matrixF[10][10];

int rowE, columnE;

int rowF, columnF;

int o,p;

printf("\nEnter the scalar: ");

scanf("%d", &scalar);

printf("\nThe scalar is: %d ", scalar);

printf("\nEnter the #rows and #cols for matrixE : ");

scanf("%d%d", &rowE, &columnE);

printf("\n\tEnter elements of Matrix E a %d x %d matrix.\n", rowE, columnE);

for (o = 0; o < rowE; o++)

{

printf("\t%d entries for row %d: ", columnE, o + 1);

for (p = 0; p < columnE; p++)

{

scanf("%d", &matrixE[o][p]);

}

}

printf("\n\t\tMatrix E\n\n");

for (o = 0; o < rowE; o++)

{

for (p = 0; p < columnE; p++)

{

printf("\t%d", matrixE[o][p]);

}

printf("\n");

}

printf("\nThe scalar multiplication between matrixE \* %d is: \n", scalar);

matrixScalarMultiply(matrixE, scalar, rowE, columnE);

break;

case 4:

int matrixG[10][10];

int matrixH[10][10];

int rowG, columnG;

int rowH, columnH;

int u,v;

printf("\nEnter the #rows and #cols for matrix G: ");

scanf("%d%d", &rowG, &columnG);

printf("Enter the #rows and #cols for matrix: ");

scanf("%d%d", &rowH, &columnH);

while (columnG != rowH)

{

printf("\n\nError! column of first matrix not equal to row of second.\n\n");

printf("\nEnter the #rows and #cols for matrix : ");

scanf("%d%d", &rowG, &columnG);

printf("Enter the #rows and #cols for matrix H: ");

scanf("%d%d", &rowH, &columnH);

}

printf("\n\tEnter elements of Matrix G a %d x %d matrix.\n", rowG, columnG);

for (u = 0; u < rowG; u++)

{

printf("\t%d entries for row %d: ", columnG, u + 1);

for (v = 0; v < columnG; v++)

{

scanf("%d", &matrixG[u][v]);

}

}

printf("\n\t\tMatrix G\n\n");

for (u = 0; u < rowG; u++)

{

for (v = 0; v < columnG; v++)

{

printf("\t%d", matrixG[u][v]);

}

printf("\n");

}

printf("\n\tEnter elements of Matrix H a %d x %d matrix.\n", rowH, columnH);

for (u = 0; u < rowH; u++

){

printf("\t%d entries for row %d: ", columnH, u + 1);

for (v = 0; v < columnH; v++)

{

scanf("%d", &matrixH[u][v]);

}

}

printf("\n\t\tMatrix H\n\n");

for (u = 0; u < rowH; u++) {

for (v = 0; v < columnH; v++){

printf("\t%d", matrixH[u][v]);

}

printf("\n");

}

matrixMultiply(matrixG, matrixH, rowG, columnG, columnH);

break;

default:

printf("\nIncorrect option! Please choose a number 1-4.");

break;

}

printf("\n\nDo you want to calculate again? Y/N\n");

scanf(" %c", &again);

again = toupper(again);

}

printf("\n\nGoodbye!\n HAVE A NICE DAY\n");

return 0;

}

void matrixAddSub(int arrayone[10][10], int arraytwo[10][10], int rows, int column, int mul){

int i, j;

int sumM[10][10];

int scaM[10][10];

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

for (j = 0; j < column; j++){

scaM[i][j] = mul \* arraytwo[i][j];

}

}

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

for (j = 0; j < column; j++){

sumM[i][j] = arrayone[i][j] + scaM[i][j];

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

}

printf("\n");

}

}

void matrixScalarMultiply(int array[10][10], int scalar, int rows, int column){

int i, j;

int scaM[10][10];

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

for (j = 0; j < column; j++){

scaM[i][j] = scalar \* array[i][j];

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

}

printf("\n");

}

}

void matrixMultiply(int arrayone[10][10], int arraytwo[10][10], int rowsA, int columnA,int columnB){

int i, j, k;

int mulM[10][10];

for (i = 0; i<rowsA; ++i)

for (j = 0; j<columnB; ++j)

{

mulM[i][j] = 0;

}

for (i = 0; i<rowsA; ++i)

for (j = 0; j<columnB; ++j)

for (k = 0; k<columnA; ++k)

{

mulM[i][j] += arrayone[i][k] \* arraytwo[k][j];

}

printf("\nOutput Matrix:\n");

for (i = 0; i<rowsA; ++i)

for (j = 0; j<columnB; ++j)

{

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

if (j == columnB - 1)

printf("\n\n");

}

}