***NAME ;* ABDULLAH**

***CMS :* 461030**

***SECTION : ME-15(B)***

***LAB TASKS***

***TASK 01;***

#include<iostream>

using namespace std;

int main() {

// Define a 3x3 matrix

int matrix[3][3] = {{1, 2, 3},

{4, 5, 6},

{7, 8, 9}};

// Calculate left diagonal sum

int leftDiagonalSum = 0;

for (int i = 0; i < 3; ++i) {

leftDiagonalSum += matrix[i][i];

}

// Calculate right diagonal sum

int rightDiagonalSum = 0;

for (int i = 0; i < 3; ++i) {

rightDiagonalSum += matrix[i][2 - i];

}

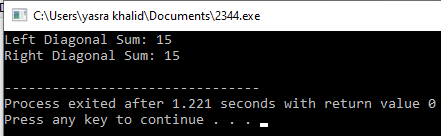
// Print the results

cout << "Left Diagonal Sum: " << leftDiagonalSum <<endl;

cout << "Right Diagonal Sum: " << rightDiagonalSum <<endl;

return 0;}

***OUTPUT***



***TASK 02;***

#include<iostream>

using namespace std;

// Define a 3x3 matrix

void addMatrices(const int matrix1[3][3], const int matrix2[3][3], int result[3][3]) {

for (int i = 0; i < 3; ++i) {

for (int j = 0; j < 3; ++j) {

result[i][j] = matrix1[i][j] + matrix2[i][j];

}

}

}

int main() {

int matrix1[3][3] = {{4,5,6}, {11,34,45}, {56,45,34}};

int matrix2[3][3] = {{56,45,34}, {11,34,45}, {4,5,6}};

int result[3][3];

addMatrices(matrix1, matrix2, result);

// Displaying the result

cout << "Resultant Matrix:" <<endl;

for (int i = 0; i < 3; ++i) {

for (int j = 0; j < 3; ++j) {

cout << result[i][j] << " ";

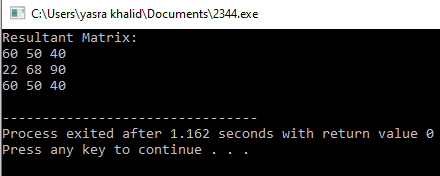
}

cout << endl;

}

return 0;}

***OUTPUT;***



***TASK 03:***

#include<iostream>

using namespace std;

void multiplyMatrices(const int matrix1[3][3], const int matrix2[3][3], int result[3][3]) {

for (int i = 0; i < 3; ++i) {

for (int j = 0; j < 3; ++j) {

result[i][j] = 0;

for (int k = 0; k < 3; ++k) {

result[i][j] += matrix1[i][k] \* matrix2[k][j];

}

}

}

}

int main() {

int matrix1[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};

int matrix2[3][3] = {{7,8,9}, {4, 5, 6}, {1, 2, 3}};

int result[3][3];

multiplyMatrices(matrix1, matrix2, result);

// Displaying the result of matrix multiplication

cout << "Matrix 1:" << endl;

for (int i = 0; i < 3; ++i) {

for (int j = 0; j < 3; ++j) {

cout << matrix1[i][j] << " ";

}

cout <<endl;

}

cout << "\nMatrix 2:" <<endl;

for (int i = 0; i < 3; ++i) {

for (int j = 0; j < 3; ++j) {

cout << matrix2[i][j] << " ";

}

cout << endl;

}

cout << "\nResultant Matrix (Matrix 1 \* Matrix 2):" <<endl;

for (int i = 0; i < 3; ++i) {

for (int j = 0; j < 3; ++j) {

cout << result[i][j] << " ";

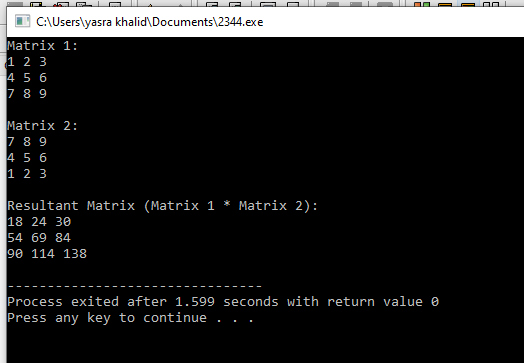
}

cout <<endl;

}

return 0;}

***OUTPUT;***



***TASK - 04;***

#include<iostream>

using namespace std;

void transposeMatrix(const int matrix[3][3], int result[3][3]) {

for (int i = 0; i < 3; ++i) {

for (int j = 0; j < 3; ++j) {

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

}

}

}

int main() {

int matrix[3][3] = {{11,23,45}, {56,67,78}, {89,90,100}};

int result[3][3];

transposeMatrix(matrix, result);

// Displaying the transposed matrix

cout << "Original Matrix:" << endl;

for (int i = 0; i < 3; ++i) {

for (int j = 0; j < 3; ++j) {

cout << matrix[i][j] << " ";

}

cout << endl;

}

cout << "Transposed Matrix:" << endl;

for (int i = 0; i < 3; ++i) {

for (int j = 0; j < 3; ++j) {

cout << result[i][j] << " ";

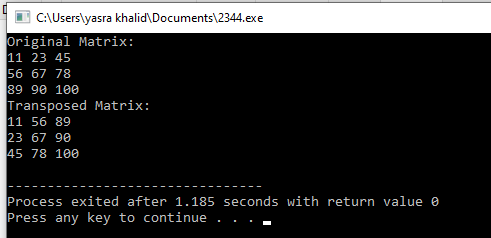
}

cout<<endl;

}

return 0;}

***oUTPUT***



***Task 5;***

#include<iostream>

using namespace std;

void printMultiplicationTable(int number, int multiplier) {

if (multiplier <= 10) {

cout << number << " \* " << multiplier << " = " << number \* multiplier << endl;

printMultiplicationTable(number, multiplier + 1);

}

}

int main() {

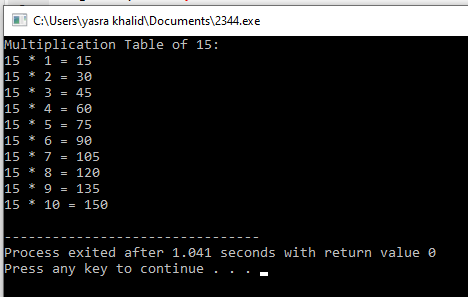
int number = 15;

cout << "Multiplication Table of " << number << ":" <<endl;

printMultiplicationTable(number, 1);

return 0;

}



***HOME TASK***

#include <iostream>

#include <cmath>

using namespace std;

// Function to calculate the determinant of a 2x2 matrix

float determinant2x2(float a, float b, float c, float d) {

return a \* d - b \* c;

}

// Function to calculate the determinant of a 3x3 matrix

float determinant3x3(float matrix[3][3]) {

return matrix[0][0] \* determinant2x2(matrix[1][1], matrix[1][2], matrix[2][1], matrix[2][2])

-

matrix[0][1] \* determinant2x2(matrix[1][0], matrix[1][2], matrix[2][0], matrix[2][2]) +

matrix[0][2] \* determinant2x2(matrix[1][0], matrix[1][1], matrix[2][0], matrix[2][1]);

}

// Function to calculate the adjoint of a 3x3 matrix

void adjoint3x3(float matrix[3][3], float adj[3][3]) {

adj[0][0] = determinant2x2(matrix[1][1], matrix[1][2], matrix[2][1], matrix[2][2]);

adj[0][1] = -determinant2x2(matrix[1][0], matrix[1][2], matrix[2][0], matrix[2][2]);

adj[0][2] = determinant2x2(matrix[1][0], matrix[1][1], matrix[2][0], matrix[2][1]);

adj[1][0] = -determinant2x2(matrix[0][1], matrix[0][2], matrix[2][1], matrix[2][2]);

adj[1][1] = determinant2x2(matrix[0][0], matrix[0][2], matrix[2][0], matrix[2][2]);

adj[1][2] = -determinant2x2(matrix[0][0], matrix[0][1], matrix[2][0], matrix[2][1]);

adj[2][0] = determinant2x2(matrix[0][1], matrix[0][2], matrix[1][1], matrix[1][2]);

adj[2][1] = -determinant2x2(matrix[0][0], matrix[0][2], matrix[1][0], matrix[1][2]);

adj[2][2] = determinant2x2(matrix[0][0], matrix[0][1], matrix[1][0], matrix[1][1]);

}

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

void inverse3x3(float matrix[3][3], float inverse[3][3]) {

float det = determinant3x3(matrix);

if (det == 0) {

cout << "Inverse does not exist as the determinant is zero." << endl;

return;

}

float adj[3][3];

adjoint3x3(matrix, adj);

// Calculate the inverse using the formula: inverse = adjoint / determinant

for (int i = 0; i < 3; ++i) {

for (int j = 0; j < 3; ++j) {

inverse[i][j] = adj[i][j] / det;

}

}

}

int main() {

float matrix[3][3];

cout << "Enter the elements of the 3x3 matrix:" << endl;

for (int i = 0; i < 3; ++i) {

for (int j = 0; j < 3; ++j) {

cin >> matrix[i][j];

}

}

float inverse[3][3];

inverse3x3(matrix, inverse);

cout << "Inverse of the matrix:" << endl;

for (int i = 0; i < 3; ++i) {

for (int j = 0; j < 3; ++j) {

cout << inverse[i][j] << " ";

}

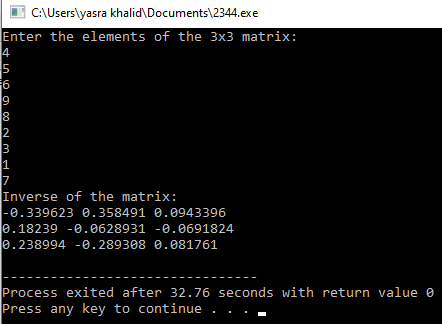
cout << endl;

}

return 0;

}

***OUTPUT***

******