LAB MANUAL 9:

# LAB TASK 1:

#include <iostream>

using namespace std;

int main() {

const int rows = 3;

const int cols = 3;

int matrix[rows][cols];

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

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

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

cout << "Enter element at position (" << i + 1 << "," << j + 1 << "): ";

cin >> matrix[i][j];

}

}

cout << "Required matrix is :"<<endl;

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

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

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

}

cout << endl;

}

int leftDiagonalSum = 0;

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

leftDiagonalSum += matrix[i][i];

}

int rightDiagonalSum = 0;

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

rightDiagonalSum += matrix[i][rows - 1 - i];

}

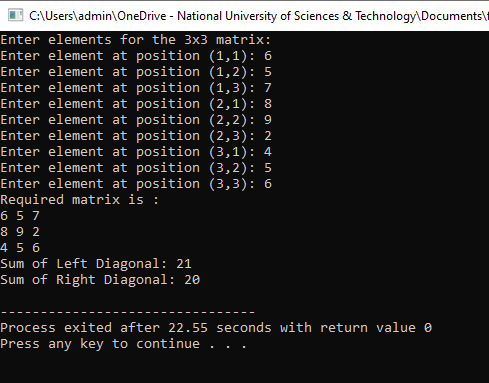
cout << "Sum of Left Diagonal: " << leftDiagonalSum << endl;

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

return 0;

}

## RESULTS:



# TASK 2:

#include <iostream>

using namespace std;

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() {

const int rows = 3;

const int cols = 3;

int matrix1[rows][cols] = {{1, 2, 3},

{4, 5, 6},

{7, 8, 9}};

int matrix2[rows][cols] = {{9, 8, 7},

{6, 5, 4},

{3, 2, 1}};

int result[rows][cols];

addMatrices(matrix1, matrix2, result)

cout << "Resultant Matrix after Addition:\n";

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

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

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

}

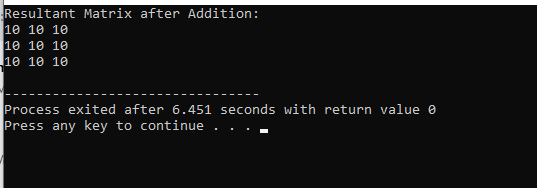
cout << endl;

}

return 0;

# }

# RESULTS:



# TASK 3:

#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() {

const int rows = 3;

const int cols = 3;

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

{4, 5, 6},

{7, 8, 9}};

int result[rows][cols];

transposeMatrix(matrix, result);

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

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

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

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

}

cout << endl;

}

cout << "Transpose Matrix:"<<endl;

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

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

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

}

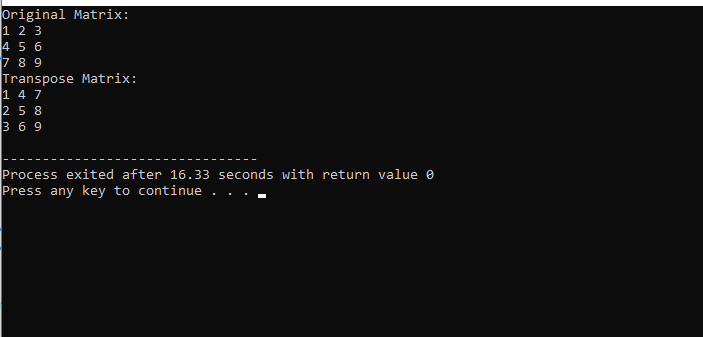
cout << endl;

}

return 0;

}

# RESULTS:



# TASK 4:

#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() {

const int rows = 3;

const int cols = 3;

int matrix1[rows][cols] = {{1, 2, 3},

{4, 5, 6},

{7, 8, 9}};

int matrix2[rows][cols] = {{9, 8, 7},

{6, 5, 4},

{3, 2, 1}};

int result[rows][cols];

multiplyMatrices(matrix1, matrix2, result);

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

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

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

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

}

cout << endl;

}

cout << "Matrix 2:"<<endl;

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

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

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

}

cout << endl;

}

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

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

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

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

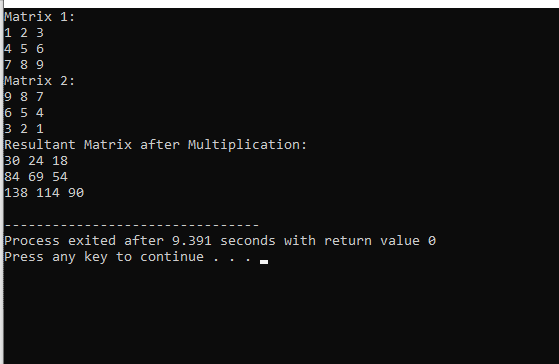
}

cout << endl;

}

return 0;

}



# TASK 5:

#include <iostream>

using namespace std;

void MultiplesOf15(int multiplier, int limit) {

if (multiplier > limit) {

return;

}

cout << "15 \* " << multiplier << " = " << 15 \* multiplier << endl;

MultiplesOf15(multiplier + 1, limit);

}

int main() {

int limit;

cout << "Enter the limit for the multiplication table of 15: ";

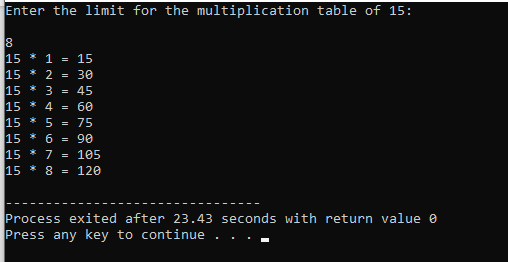
cin >> limit;

MultiplesOf15(1, limit);

return 0;

}

# RSULTS:



HOME TASK 1:

#include <iostream>

#include <cmath>

using namespace std;

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

return a \* d - b \* c;

}

double determinant(double mat[3][3]) {

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

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

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

}

void transpose(double mat[3][3], double result[3][3]) {

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

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

result[i][j] = mat[j][i];

}

void cofactor(double mat[3][3], double result[3][3]) {

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

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

int sign = ((i + j) % 2 == 0) ? 1 : -1;

double temp[2][2];

int row = 0, col = 0;

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

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

if (k != i && l != j) {

temp[row][col++] = mat[k][l];

if (col == 2) {

col = 0;

row++;

}

}

}

}

result[i][j] = sign \* determinant2x2(temp[0][0], temp[0][1], temp[1][0], temp[1][1]);

}

}

}

bool inverse(double mat[3][3], double result[3][3]) {

double det = determinant(mat);

if (det == 0) {

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

return false;

}

double adj[3][3];

cofactor(mat, adj);

transpose(adj, result);

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

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

result[i][j] /= det;

return true;

}

int main() {

double mat[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 >> mat[i][j];

double inv[3][3];

if (inverse(mat, inv)) {

cout << "Inverse Matrix:" << endl;

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

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

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

cout << endl;

}

}

return 0;

}

## RESULTS:

