**Name : Ahmed Masood**

**Me 15 c**

**Cms : 465073**

Is common in all

**Task 1**

#include<iostream>

using namespace std;

is common in all

int main(){

int arr[3][3];

int leftsum=0;

int rightsum=0;

cout<<"enter the elements of the 2d array";

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

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

cin>>arr[i][j];

}

}

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

leftsum+=arr[i][i];

}

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

rightsum+=arr[i][2-i];

}

cout<<"the left diagnol sum is as follows"<<" "<<leftsum<<endl<<"the right diagnol sum is as follows"<<" "<<rightsum;

}

**Task 2**

void addmatrices(int matrix1[3][3] , 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 result[3][3];

int matrix1[3][3];

int matrix2[3][3];

cout<<"enter elements of array 1";

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

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

cin>>matrix1[i][j];

}

}

cout<<"enter elements of array 2";

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

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

cin>>matrix2[i][j];

}

}

addmatrices(matrix1,matrix2,result);

cout<<"sum of matrices"<<endl;

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

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

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

}

cout<<endl;

}

return 0;

}

**Task 3**

void transpose(int matrix[3][3])

{

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

for (int j=i+1;j<3;j++){

int temp= matrix[i][j];

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

matrix[j][i]=temp;

}

}

}

int main(){

int matrix[3][3];

cout<<"enter the elements of the array"<<endl;

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

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

cin>>matrix[i][j];

}

}cout<<endl;

cout<<"the transpose of the matrix is as follows"<<endl;

transpose(matrix);

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

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

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

}cout<<endl;

}

}

**Task 4**

void multiplication(int matrix1[3][3],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],matrix2[3][3],result[3][3];

cout<<"enter the elements of the array 1"<<endl;

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

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

cin>> matrix1[i][j];

}

}cout<<"enter bthe elements of 2nd array"<<endl;

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

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

cin>> matrix2[i][j];

}

}

cout<<"the result is as follows"<<endl;

multiplication(matrix1,matrix2,result);

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

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

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

}cout<<endl;

}cout<<endl;

}

**Task 5**

void printMultiplicationTable(int num, int multiplier) {

if (multiplier > 10) {

return;

}

cout << num << " \* " << multiplier << " = " << num \* multiplier << std::endl;

printMultiplicationTable(num, multiplier + 1);

}

int main() {

int tableNumber = 15;

cout << "Multiplication table of " << tableNumber << ":" << std::endl;

printMultiplicationTable(tableNumber, 1);

return 0;

}

**Hometask**

float det(float mat[3][3]) {

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

mat[0][1] \* (mat[1][0] \* mat[2][2] - mat[1][2] \* mat[2][0]) +

mat[0][2] \* (mat[1][0] \* mat[2][1] - mat[1][1] \* mat[2][0]);

}

void adjoint(float mat[3][3], float adj[3][3]) {

adj[0][0] = mat[1][1] \* mat[2][2] - mat[1][2] \* mat[2][1];

adj[0][1] = mat[0][2] \* mat[2][1] - mat[0][1] \* mat[2][2];

adj[0][2] = mat[0][1] \* mat[1][2] - mat[0][2] \* mat[1][1];

adj[1][0] = mat[1][2] \* mat[2][0] - mat[1][0] \* mat[2][2];

adj[1][1] = mat[0][0] \* mat[2][2] - mat[0][2] \* mat[2][0];

adj[1][2] = mat[0][2] \* mat[1][0] - mat[0][0] \* mat[1][2];

adj[2][0] = mat[1][0] \* mat[2][1] - mat[1][1] \* mat[2][0];

adj[2][1] = mat[0][1] \* mat[2][0] - mat[0][0] \* mat[2][1];

adj[2][2] = mat[0][0] \* mat[1][1] - mat[0][1] \* mat[1][0];

}

void inverse(float mat[3][3]) {

float determinant = det(mat);

if (determinant == 0) {

cout << "The matrix is singular, and its inverse does not exist." << std::endl;

return;

}

float adj[3][3];

adjoint(mat, adj);

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

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

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

cout << adj[i][j] / determinant << " ";

}

cout << std::endl;

}

}

int main() {

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

inverse(matrix);

return 0;

}