#include <iostream>

using namespace std;

// Function to partition the array

int partition(int arr[], int low, int high) {

int pivot = arr[high]; // Choose the last element as pivot

int i = low - 1; // Index of smaller element

for(int j = low; j < high; j++) {

// If current element is smaller than or equal to pivot

if(arr[j] <= pivot) {

i++;

swap(arr[i], arr[j]); // Swap elements

}

}

swap(arr[i + 1], arr[high]); // Place pivot in correct position

return i + 1;

}

// Quick Sort function

void quickSort(int arr[], int low, int high) {

if(low < high) {

// Find pivot element such that elements left are smaller, right are larger

int pi = partition(arr, low, high);

// Recursively sort elements before and after partition

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}

// Function to print the array

void printArray(int arr[], int size) {

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

cout << arr[i] << " ";

cout << endl;

}

// Main function

int main() {

int n;

cout << "Enter number of elements: ";

cin >> n;

int arr[n];

cout << "Enter elements:\n";

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

cin >> arr[i];

cout << "Original array:\n";

printArray(arr, n);

quickSort(arr, 0, n - 1);

cout << "Sorted array:\n";

printArray(arr, n);

return 0;

}

#include <iostream>

using namespace std;

const int MAX = 2; // For simplicity, using 2x2 matrices

void strassen(int A[MAX][MAX], int B[MAX][MAX], int C[MAX][MAX]) {

int M1 = (A[0][0] + A[1][1]) \* (B[0][0] + B[1][1]);

int M2 = (A[1][0] + A[1][1]) \* B[0][0];

int M3 = A[0][0] \* (B[0][1] - B[1][1]);

int M4 = A[1][1] \* (B[1][0] - B[0][0]);

int M5 = (A[0][0] + A[0][1]) \* B[1][1];

int M6 = (A[1][0] - A[0][0]) \* (B[0][0] + B[0][1]);

int M7 = (A[0][1] - A[1][1]) \* (B[1][0] + B[1][1]);

C[0][0] = M1 + M4 - M5 + M7;

C[0][1] = M3 + M5;

C[1][0] = M2 + M4;

C[1][1] = M1 - M2 + M3 + M6;

}

int main() {

int A[MAX][MAX], B[MAX][MAX], C[MAX][MAX];

cout << "Enter elements of Matrix A (2x2):\n";

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

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

cin >> A[i][j];

cout << "Enter elements of Matrix B (2x2):\n";

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

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

cin >> B[i][j];

strassen(A, B, C);

cout << "Resultant Matrix C:\n";

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

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

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

cout << endl;

}

return 0;

}

