#include <iostream>

#include <vector>

#include <cstdlib>

#include <ctime>

#include <chrono>

#include <algorithm>

using namespace std;

// Function to partition the array for the deterministic Quick Sort

int deterministicPartition(vector<int>& arr, int low, int high) {

int pivot = arr[high];

int i = (low - 1);

for (int j = low; j <= high - 1; j++) {

if (arr[j] < pivot) {

i++;

swap(arr[i], arr[j]);

}

}

swap(arr[i + 1], arr[high]);

return (i + 1);

}

// Deterministic Quick Sort algorithm

void deterministicQuickSort(vector<int>& arr, int low, int high) {

if (low < high) {

int pivot = deterministicPartition(arr, low, high);

deterministicQuickSort(arr, low, pivot - 1);

deterministicQuickSort(arr, pivot + 1, high);

}

}

// Function to choose a random pivot for the randomized Quick Sort

int randomizedPartition(vector<int>& arr, int low, int high) {

int randomIndex = low + rand() % (high - low + 1);

swap(arr[randomIndex], arr[high]);

return deterministicPartition(arr, low, high);

}

// Randomized Quick Sort algorithm

void randomizedQuickSort(vector<int>& arr, int low, int high) {

if (low < high) {

int pivot = randomizedPartition(arr, low, high);

randomizedQuickSort(arr, low, pivot - 1);

randomizedQuickSort(arr, pivot + 1, high);

}

}

int main() {

srand(time(0)); // Initialize random seed

int n = 10000; // Change n to the desired array size

vector<int> deterministicArr(n);

vector<int> randomizedArr(n);

// Generate random input data

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

int randomValue = rand() % n;

deterministicArr[i] = randomizedArr[i] = randomValue;

}

cout << "Array of size " << n << " created with random values." << endl;

// Analyze and measure deterministic Quick Sort

auto start = chrono::high\_resolution\_clock::now();

deterministicQuickSort(deterministicArr, 0, n - 1);

auto end = chrono::high\_resolution\_clock::now();

auto deterministicTime = chrono::duration\_cast<chrono::microseconds>(end - start).count();

// Analyze and measure randomized Quick Sort

start = chrono::high\_resolution\_clock::now();

randomizedQuickSort(randomizedArr, 0, n - 1);

end = chrono::high\_resolution\_clock::now();

auto randomizedTime = chrono::duration\_cast<chrono::microseconds>(end - start).count();

// Check if the arrays are sorted correctly

bool isSorted = (deterministicArr == randomizedArr);

cout << "Deterministic Quick Sort time: " << deterministicTime << " microseconds." << endl;

cout << "Randomized Quick Sort time: " << randomizedTime << " microseconds." << endl;

cout << "Arrays are sorted correctly: " << (isSorted ? "Yes" : "No") << endl;

for(int i=0;i<deterministicArr.size();i++){

cout<<deterministicArr[i]<<" ";

}

return 0;

}