6. DAA: Write a program for analysis of quick sort by using

deterministic and randomized variant. With explanation and

viva questions in c++

#include <iostream>

#include <cstdlib>

#include <ctime>

using namespace std;

// Function to swap two elements

void swap(int &a, int &b) {

int temp = a;

a = b;

b = temp;

}

// Deterministic partition function (using last element as pivot)

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

int pivot = arr[high]; // pivot

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

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

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

if (low < high) {

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

// Recursively sort elements before partition and after

partition

deterministicQuickSort(arr, low, pi - 1);

deterministicQuickSort(arr, pi + 1, high);

}

}

// Randomized partition function

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

// Generate a random index between low and high

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

swap(arr[randomIndex], arr[high]); // Swap with the last

element

return deterministicPartition(arr, low, high);

}

// Randomized Quick Sort

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

if (low < high) {

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

// Recursively sort elements before partition and after

partition

randomizedQuickSort(arr, low, pi - 1);

randomizedQuickSort(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 to analyze both algorithms

int main() {

srand(time(0)); // Seed for random number generation

int n;

cout << "Enter the number of elements: ";

cin >> n;

int \*arr1 = new int[n];

int \*arr2 = new int[n];

cout << "Enter the elements:\n";

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

cin >> arr1[i];

arr2[i] = arr1[i]; // Copy elements to the second array for

randomized quick sort

}

// Analyze deterministic quick sort

cout << "\nDeterministic Quick Sort:\n";

deterministicQuickSort(arr1, 0, n - 1);

printArray(arr1, n);

// Analyze randomized quick sort

cout << "\nRandomized Quick Sort:\n";

randomizedQuickSort(arr2, 0, n - 1);

printArray(arr2, n);

// Clean up dynamically allocated memory

delete[] arr1;

delete[] arr2;

return 0;

}