#include <iostream>

#include <vector>

#include <omp.h>

#include <chrono>

using namespace std;

using namespace std::chrono;

void printArray(const vector<int>& arr) {

for (int num : arr) {

cout << num << " ";

}

cout << endl;

}

void bubbleSortSequential(vector<int>& arr) {

int n = arr.size();

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

if (arr[j] > arr[j + 1]) {

swap(arr[j], arr[j + 1]);

}

}

}

}

void bubbleSortParallel(vector<int>& arr) {

int n = arr.size();

#pragma omp parallel for

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

if (arr[j] > arr[j + 1]) {

swap(arr[j], arr[j + 1]);

}

}

}

}

int main() {

int n;

cout << "Enter number of elements: ";

cin >> n;

vector<int> arr(n);

cout << "Enter elements: ";

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

cin >> arr[i];

}

vector<int> arrCopy = arr;

// Time for Sequential Bubble Sort

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

bubbleSortSequential(arrCopy);

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

auto duration = duration\_cast<milliseconds>(end - start);

cout << "Sequential Bubble Sort Time: " << duration.count() << " milliseconds\n";

cout << "Sorted Array (Bubble Sort - Sequential): ";

printArray(arrCopy);

arrCopy = arr;

// Time for Parallel Bubble Sort

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

bubbleSortParallel(arrCopy);

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

duration = duration\_cast<milliseconds>(end - start);

cout << "Parallel Bubble Sort Time: " << duration.count() << " milliseconds\n";

cout << "Sorted Array (Bubble Sort - Parallel): ";

printArray(arrCopy);

return 0;

}