#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 merge(vector<int>& arr, int left, int mid, int right) {

int n1 = mid - left + 1;

int n2 = right - mid;

vector<int> L(n1), R(n2);

for (int i = 0; i < n1; i++)

L[i] = arr[left + i];

for (int j = 0; j < n2; j++)

R[j] = arr[mid + 1 + j];

int i = 0, j = 0, k = left;

while (i < n1 && j < n2) {

if (L[i] <= R[j]) {

arr[k] = L[i];

i++;

} else {

arr[k] = R[j];

j++;

}

k++;

}

while (i < n1) {

arr[k] = L[i];

i++;

k++;

}

while (j < n2) {

arr[k] = R[j];

j++;

k++;

}

}

void mergeSortSequential(vector<int>& arr, int left, int right) {

if (left < right) {

int mid = left + (right - left) / 2;

mergeSortSequential(arr, left, mid);

mergeSortSequential(arr, mid + 1, right);

merge(arr, left, mid, right);

}

}

void mergeSortParallel(vector<int>& arr, int left, int right) {

if (left < right) {

int mid = left + (right - left) / 2;

#pragma omp parallel sections

{

#pragma omp section

mergeSortParallel(arr, left, mid);

#pragma omp section

mergeSortParallel(arr, mid + 1, right);

}

merge(arr, left, mid, right);

}

}

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 Merge Sort

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

mergeSortSequential(arrCopy, 0, n - 1);

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

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

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

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

printArray(arrCopy);

arrCopy = arr;

// Time for Parallel Merge Sort

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

mergeSortParallel(arrCopy, 0, n - 1);

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

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

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

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

printArray(arrCopy);

return 0;

}