#include <iostream>

#include <vector>

#include <algorithm>

#include <omp.h>

using namespace std;

// Parallel bubble sort implementation

void parallel\_bubble\_sort(vector<int>& arr) {

int n = arr.size();

bool swapped = true;

while (swapped) {

swapped = false;

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

if (arr[i - 1] > arr[i]) {

swap(arr[i - 1], arr[i]);

swapped = true;

}

}

}

}

// Parallel merge sort implementation

void parallel\_merge\_sort(vector<int>& arr) {

if (arr.size() > 1) {

vector<int> left(arr.begin(), arr.begin() + arr.size() / 2);

vector<int> right(arr.begin() + arr.size() / 2, arr.end());

#pragma omp parallel sections

{

#pragma omp section

parallel\_merge\_sort(left);

#pragma omp section

parallel\_merge\_sort(right);

}

merge(left.begin(), left.end(), right.begin(), right.end(), arr.begin());

}

}

void show(int op, vector<int>& arr){

vector<int> copy = arr; string str="", name="";

switch(op){

case 0: name="Original"; str=" without"; break;

case 1: name="Sequential"; str="bubble";

sort(copy.begin(), copy.end()); break;

case 2: name="Parallel"; str=" bubble";

parallel\_bubble\_sort(copy); break;

case 3: name="Sequential"; str=" merge";

stable\_sort(copy.begin(), copy.end()); break;

case 4: name="Parallel"; str=" merge";

parallel\_merge\_sort(copy); break;

}

cout << name << " " << str << " sort : ";

for (const auto& num : copy) cout << num << " ";

cout << endl;

}

int main() {

vector<int> arr{ 4, 2, 6, 8, 1, 3, 9, 5, 7 };

for(int i=0; i<5; i++) show(i, arr);

return 0;

}

// • Output:

// Original without sort : 4 2 6 8 1 3 9 5 7

// Sequential bubble sort : 1 2 3 4 5 6 7 8 9

// Parallel bubble sort : 1 2 3 4 5 6 7 8 9

// Sequential merge sort : 1 2 3 4 5 6 7 8 9

// Parallel merge sort : 1 2 3 4 5 6 7 8 9