#include <iostream>

#include <vector>

#include <algorithm>

#include <omp.h>

using namespace std;

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

int n = arr.size(); bool swapped = true;

while (swapped) { swapped = false;

#pragma omp parallel for shared(arr)

for (int i = 1; i < n; ++i) if (arr[i - 1] > arr[i]) {

swap(arr[i - 1], arr[i]); swapped = true;

}

}

}

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

if (arr.size() > 1) { vector<int>

l(a.begin(),a.begin()+a.size()/2),r(a.begin()+a.size()/2,a.end());

#pragma omp parallel sections

{

#pragma omp section

parallel\_merge\_sort(l);

#pragma omp section

parallel\_merge\_sort(r);

} merge(l.begin(), l.end(), r.begin(), r.end(), arr.begin());

}

}

void show(int op, vector<int>& arr){ vector<int> c = arr; string s="", n="";

switch(op){

case 0: n="Original" ; s=" without"; break;

case 1: n="Sequential"; s="bubble" ; sort(c.begin(), c.end()); break;

case 2: n="Parallel" ; s=" bubble"; parallel\_bubble\_sort(c); break;

case 3: n="Sequential"; s=" merge" ; stable\_sort(c.begin(), c.end()); break;

case 4: n="Parallel" ; s=" merge" ; parallel\_merge\_sort(c); break;

} cout << n << " " << s << " sort : ";

for (const auto& num : c) 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;

}