#include <iostream>

#include <vector>

#include <omp.h>

void merge(std::vector<int>& arr, int left, int middle, int right) {

std::vector<int> temp(right - left + 1);

int i = left;

int j = middle + 1;

int k = 0;

while (i <= middle && j <= right) {

if (arr[i] <= arr[j])

temp[k++] = arr[i++];

else

temp[k++] = arr[j++];}

while (i <= middle)

temp[k++] = arr[i++];

while (j <= right)

temp[k++] = arr[j++];

for (i = left, k = 0; i <= right; i++, k++)

arr[i] = temp[k];}

void parallelMergeSort(std::vector<int>& arr, int left, int right) {

if (left < right) {

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

#pragma omp parallel sections

{

#pragma omp section

parallelMergeSort(arr, left, middle);

#pragma omp section

parallelMergeSort(arr, middle + 1, right);

}

merge(arr, left, middle, right);}}

int main() {

std::vector<int> arr = {5, 2, 8, 12, 1, 6, 3, 9};

parallelMergeSort(arr, 0, arr.size() - 1);

for (int num : arr)

std::cout << num << " ";

std::cout << std::endl;

return 0;}