#include <iostream>

#include <omp.h>

using namespace std;

// Function to merge two subarrays

void merge(int a[], int i1, int j1, int i2, int j2) {

int temp[1000];

int i, j, k;

i = i1;

j = i2;

k = 0;

while (i <= j1 && j <= j2) {

if (a[i] < a[j]) {

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

} else {

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

}

}

while (i <= j1) {

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

}

while (j <= j2) {

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

}

for (i = i1, j = 0; i <= j2; i++, j++) {

a[i] = temp[j];

}

}

// Parallel Merge Sort function

void mergesort(int a[], int i, int j) {

int mid;

if (i < j) {

mid = (i + j) / 2;

#pragma omp parallel sections

{

#pragma omp section

{

mergesort(a, i, mid);

}

#pragma omp section

{

mergesort(a, mid + 1, j);

}

}

merge(a, i, mid, mid + 1, j);

}

}

int main() {

int \*a, n, i;

cout << "\nEnter total number of elements: ";

cin >> n;

a = new int[n];

cout << "\nEnter elements:\n";

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

cin >> a[i];

}

// Start parallel sorting

mergesort(a, 0, n - 1);

cout << "\nSorted array is:\n";

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

cout << a[i] << "\n";

}

delete[] a; // Free the allocated memory

return 0;

}