```
#include <iostream>
#include <vector>
#include <omp.h>
using namespace std;
// 1. Parallel Bubble Sort
void parallelBubbleSort(vector<int>& arr) {
int n = arr.size();
#pragma omp parallel
for (int i = 0; i < n - 1; i++) {
#pragma omp for
for (int j = 0; j < n - i - 1; j++) {
if (arr[j] > arr[j + 1]) {
swap(arr[j], arr[j + 1]);
}
}
}
}
// 2. Parallel Merge Sort
void merge(vector<int>& arr, int left, int mid, int right) {
vector<int> temp(right - left + 1);
int i = left, j = mid + 1, k = 0;
while (i <= mid && j <= right) {
temp[k++] = (arr[i] < arr[j]) ? arr[i++] : arr[j++];
}
while (i <= mid) {
temp[k++] = arr[i++];
}
while (j <= right) {
temp[k++] = arr[j++];
}
for (int m = 0; m < k; m++) {
```

```
arr[left + m] = temp[m];
}
}
void parallelMergeSort(vector<int>& arr, int left, int right) {
if (left >= right) {
return;
}
int mid = left + (right - left) / 2;
#pragma omp parallel sections
{
#pragma omp section
parallelMergeSort(arr, left, mid);
#pragma omp section
parallelMergeSort(arr, mid + 1, right);
}
merge(arr, left, mid, right);
}
int main() {
int n;
cout << "Enter number of elements: "; cin >> n;
vector<int> arr(n), arr2;
cout << "Enter elements: ";
for (int& x : arr) {
cin >> x;
}
arr2 = arr;
double start, end;
start = omp_get_wtime();
parallelBubbleSort(arr);
end = omp_get_wtime();
cout << "Parallel Bubble Sort: ";
```

```
for (int x : arr) {
cout << x << " ";
}
cout << "\nTime: " << (end - start) << " sec\n";
start = omp_get_wtime();
parallelMergeSort(arr2, 0, n - 1);
end = omp_get_wtime();
cout << "Parallel Merge Sort: ";
for (int x : arr2) {
cout << x << " ";
}
cout << "\nTime: " << (end - start) << " sec\n";
}
```