Hpc parallel merge sort

```
#include <iostream>
#include <omp.h>

using namespace std;

void merge(int a[], int i1, int j1, int i2, int j2) {
   int temp[1000];
   int 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];
   }
}

void mergesort(int a[], int i, int j) {
   if (i < j) {
       int 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;
   cout << "Enter total number of elements: ";
   cin >> n;
   a = new int[n];
   cout << "Enter elements: ";
   for (int i = 0; i < n; i++) {
       cin >> a[i];
   }
```

```
    double start_time = omp_get_wtime();
    #pragma omp parallel
    {
        #pragma omp single
        {
            mergesort(a, 0, n - 1);
        }
    }
    double end_time = omp_get_wtime();

    cout << "Sorted array: ";
    for (int i = 0; i < n; i++) {
        cout << a[i] << " ";
    }
    cout << "\nTime taken: " << end_time - start_time << " seconds" << endl;

    delete[] a;
    return 0;
}
```