```
#include <iostream>
#include <omp.h>
using namespace std;

void mergesort(int a[], int i, int j);
void merge(int a[], int i1, int j1, int i2, int j2);

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);
    }
}

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];
}

int main()
{
    int *a, n, i;
    double start_time, end_time, seq_time, par_time;

    cout << "\n enter total no of elements=>";
    cin >> n;
    a = new int[n];

    cout << "\n enter elements=>";
    for (i = 0; i < n; i++)
    {
        cin >> a[i];
    }

    // Sequential algorithm
    start_time = omp_get_wtime();
```

```
    mergesort(a, 0, n - 1);
    end_time = omp_get_wtime();
    seq_time = end_time - start_time;
    cout << "\nSequential Time: " << seq_time << endl;

    // Parallel algorithm
    start_time = omp_get_wtime();
    #pragma omp parallel
    {
        #pragma omp single
        mergesort(a, 0, n - 1);
    }
    end_time = omp_get_wtime();
    par_time = end_time - start_time;
    cout << "\nParallel Time: " << par_time << endl;

    cout << "\n sorted array is=>";
    for (i = 0; i < n; i++)
    {
        cout << "\n" << a[i];
    }

    delete[] a;

    return 0;
}

#OUTPUT:-
enter total no of elements=>5
enter elements=>5 3 1 4 2

Sequential Time: 0.00016079
Parallel Time: 0.000197933

sorted array is=>
1
2
3
4
5
```