Parallel_Merge.cpp

```
#include<iostream>
#include<stdlib.h>
#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)
{
   int mid;
   if (i < j) {
      mid = (i + j) / 2;
      // Parallelize the two recursive merge sort calls
      #pragma omp parallel
      {
         // Print the number of threads used in this parallel region
         #pragma omp single
         {
            cout << "\nNumber of threads in mergesort: " << omp_get_num_threads() << endl;
         }
         #pragma omp sections
         {
            #pragma omp section
            {
               mergesort(a, i, mid);  // Sort the first half
            }
            #pragma omp section
            {
               mergesort(a, mid + 1, j);  // Sort the second half
```

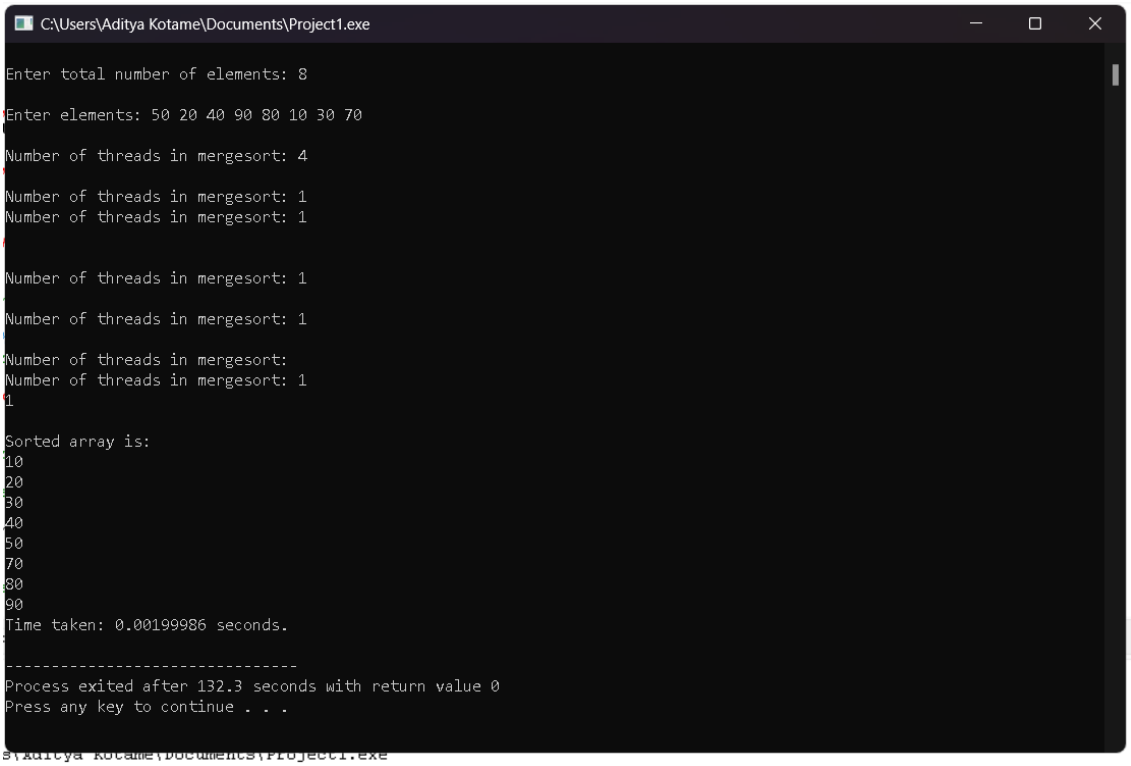
```
            }
          }
        }
        merge(a, i, mid, mid + 1, j);  // Merge the sorted halves
    }
}
void merge(int a[], int i1, int j1, int i2, int j2)
{
    int temp[1000];
    int i, j, k;
    i = i1;
    j = i2;
    k = 0;

    // Merge the two sorted halves into the temp array
    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++];
    }
    // Copy the sorted elements back into the original array
    for (i = i1, j = 0; i <= j2; i++, j++) {
        a[i] = temp[j];
```

```
    }
}
int main()
{
    int *a, n, i;
    cout << "\nEnter total number of elements: ";
    cin >> n;
    a = new int[n];

    cout << "\nEnter elements: ";
    for (i = 0; i < n; i++) {
        cin >> a[i];
    }
    // Start timer
    double start_time = omp_get_wtime();
    mergesort(a, 0, n - 1);  // Perform merge sort
    // Stop timer
    double end_time = omp_get_wtime();
    cout << "\nSorted array is: ";
    for (i = 0; i < n; i++) {
        cout << "\n" << a[i];
    }
    // Print time taken for sorting
    cout << "\nTime taken: " << end_time - start_time << " seconds." << endl;
    delete[] a;  // Free dynamically allocated memory
    return 0;
}
```

Output:


C:\Users\Aditya Kotame\Documents\Project1.exe
Enter total number of elements: 8
Enter elements: 50 20 40 90 80 10 30 70
Number of threads in mergesort: 4
Number of threads in mergesort: 1
Number of threads in mergesort: 1
Number of threads in mergesort: 1
Number of threads in mergesort: 1
Number of threads in mergesort:
Number of threads in mergesort: 1
1
Sorted array is:
10
20
30
40
50
70
80
90
Time taken: 0.00199986 seconds.
Process exited after 132.3 seconds with return value 0
Press any key to continue . . .