Practical 2 : Merge Sort

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

const int threshold = 1000; // Set your desired threshold here


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

void mergesort(int a[], int i, int j)
{
  int mid;
  if (i < j)
  {
    mid = (i + j) / 2;

    if ((j - i + 1) <= threshold)
    {
      mergesort_seq(a, i, mid);
      mergesort_seq(a, mid + 1, j);
    }
    else
    {
      #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 mergesort_seq(int a[], int i, int j)
{
    int mid;
    if (i < j)
    {
        mid = (i + j) / 2;

        mergesort(a, i, mid);
        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 size = (j1 - i1 + 1) + (j2 - i2 + 1);
    int* temp = new int[size];
    int i=i1;
    int j=i2;
    int 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++)
    {
```

```cpp
            a[i]=temp[j];
        }

        delete[] temp; // Free the dynamically allocated memory
    }

    int main()
    {
        cout<<"This is Atharva Pingale's Code";
        cout<<"\nPractical 2 : Merge Sort\n";
        int *a,*b;
        long int n,i;
        double start_time, end_time, seq_time, parallel_time;

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

        for(i=0;i<n;i++)
        {
            int random_value = (rand()%(9999999 - 9999 + 1));
            a[i] = random_value;
            b[i] = random_value;
        }

        // Sequential algorithm
        start_time = omp_get_wtime();
        mergesort_seq(a, 0, n-1);
        end_time = omp_get_wtime();
        seq_time = end_time - start_time;

        cout<<"\nFirst 20 elements of sorted array ( Sequential algorithm ) : \n";
        for(i=0;i<20;i++)
        {
            cout<<a[i]<<", ";
        }

        // Parallel algorithm
        start_time = omp_get_wtime();
        #pragma omp parallel
        {
            #pragma omp single
            {
                mergesort(b, 0, n-1);
            }
        }
```

```cpp
        end_time = omp_get_wtime();
        parallel_time = end_time - start_time;

        cout<<"\nFirst 20 elements of sorted array ( Parallel algorithm ) : \n";
        for(i=0;i<20;i++)
        {
            cout<<b[i]<<", ";
        }
        cout << "\n\nSequential Time: " << seq_time;
        cout << "\n\nParallel Time: " << parallel_time;

        delete[] a;
        delete[] b;

        return 0;
}
```

Output :

```
athar@LAPTOP-U0997R48 MINGW64 /d/GitHub/BE-8th-Semester/hpc_practicals (main)
$ g++ -fopenmp merge_sort.cpp -o merge_sort

athar@LAPTOP-U0997R48 MINGW64 /d/GitHub/BE-8th-Semester/hpc_practicals (main)
$ ./merge_sort
This is Atharva Pingale's Code
Practical 2 : Merge Sort

Enter total no of elements=>99999

First 20 elements of sorted array ( Sequential algorithm ) :
0, 0, 1, 1, 1, 1, 1, 1, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 6, 6,
First 20 elements of sorted array ( Parallel algorithm ) :
0, 0, 1, 1, 1, 1, 1, 1, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 6, 6,

Sequential Time: 0.0309999

Parallel Time: 0.023
athar@LAPTOP-U0997R48 MINGW64 /d/GitHub/BE-8th-Semester/hpc_practicals (main)
$ []
```

```
athar@LAPTOP-U0997R48 MINGW64 /d/GitHub/BE-8th-Semester/hpc_practicals (main)
$ g++ -fopenmp merge_sort.cpp -o merge_sort

athar@LAPTOP-U0997R48 MINGW64 /d/GitHub/BE-8th-Semester/hpc_practicals (main)
$ ./merge_sort
This is Atharva Pingale's Code
Practical 2 : Merge Sort

Enter total no of elements=>99999

First 20 elements of sorted array ( Sequential algorithm ) :
9999, 9999, 10000, 10000, 10000, 10000, 10000, 10000, 10002, 10002, 10002, 10003, 10003, 10003, 10003, 10003, 10004, 10004, 10005, 10005,
First 20 elements of sorted array ( Parallel algorithm ) :
9999, 9999, 10000, 10000, 10000, 10000, 10000, 10000, 10002, 10002, 10002, 10003, 10003, 10003, 10003, 10003, 10004, 10004, 10005, 10005,

Sequential Time: 0.016

Parallel Time: 0.00999999
athar@LAPTOP-U0997R48 MINGW64 /d/GitHub/BE-8th-Semester/hpc_practicals (main)
$ |
```