HPC Assignment-03

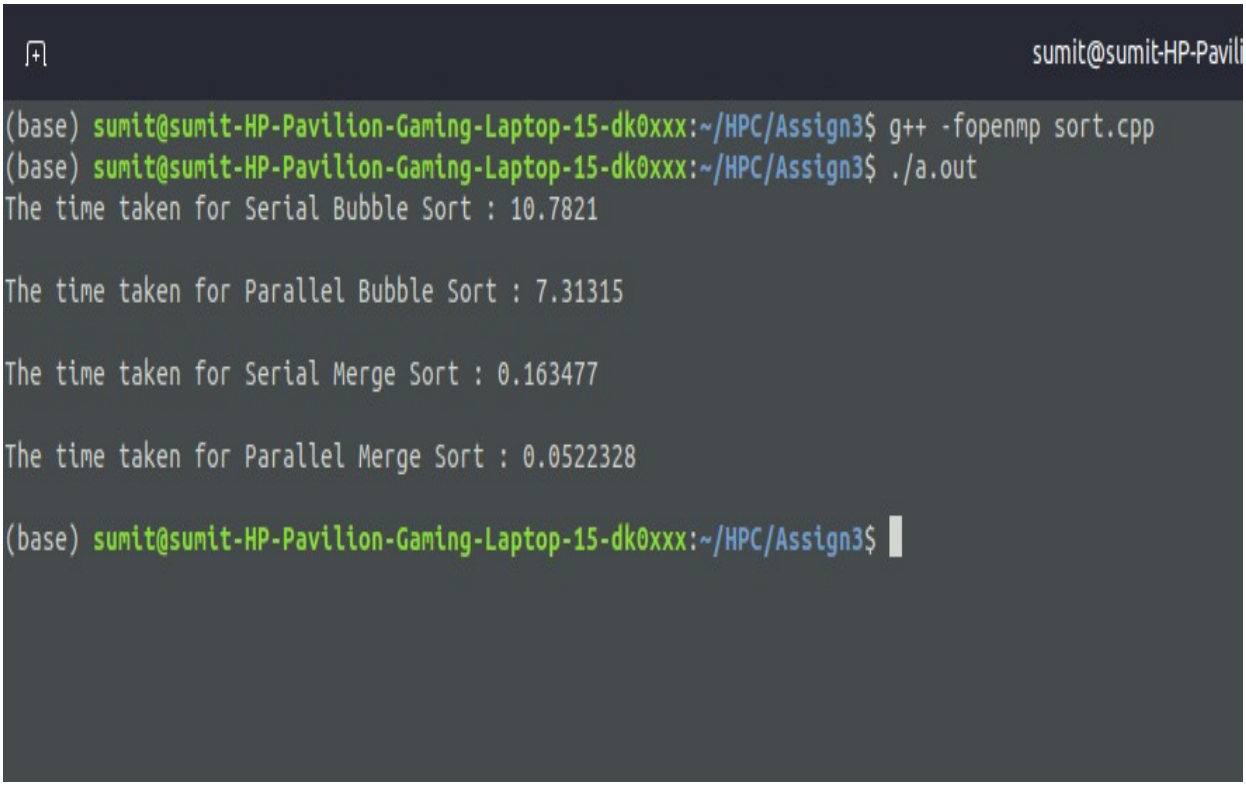
Code:

```
#include <iostream>
#include<omp.h>
#include<stdio.h>
#include<stdlib.h>
#define N 100000

using namespace std;
int a[N];

void swap(int *a,int *b) {
        int temp =  *a;
        *a = *b;
        *b = temp;
}

void seqBubbleSort(int *a) {
        int arr[N];
        double start, end;
        for(int i=0;i<N;i++) {
                arr[i]=a[i];
        }
        start = omp_get_wtime();
        for(int i=0;i<N;i++) {
                for(int j=i;j<N; j+=2) {
                        if(arr[j]>arr[j+1]) {
                                swap(&arr[j],&arr[j+1]);
                        }
                }
        }
        end = omp_get_wtime();
   cout<<"The time taken for Serial Bubble Sort : "<<(end-start)<<endl;
   cout<<endl;
}

void parBubbleSort(int *a) {
        int arr[N];
        double start, end;
        for(int i=0;i<N;i++) {
                arr[i]=a[i];
        }
        start = omp_get_wtime();
        for(int i=0; i<N-1; i++) {
      int first = i%2;
      #pragma omp parallel for num_threads(4)
      for(int j = first; j < N-1; j+=2){
                        if(arr[j] > arr[j+1]) {
                        swap(&arr[j], &arr[j+1]);
                }
```

```
            }
        }
        end = omp_get_wtime();
        cout<<"The time taken for Parallel Bubble Sort : "<<(end-start)<<endl;
        cout<<endl;
}

void merge(int *arr, int start, int mid, int end){
    int len = (end - start) + 1, temp[len], cur = 0, i = start, j = mid+1;

    while(i <= mid && j <= end){
        if(arr[i] <= arr[j]){
            temp[cur] = arr[i];
            i++;
        }
        else{
            temp[cur] = arr[j];
             j++;
        }
        cur++;
    }
    while(i <= mid){
        temp[cur] = arr[i];
        i++;
        cur++;
    }
    while(j <= end){
        temp[cur] = arr[i];
        j++;
        cur++;
    }
    cur = 0;
    for(i=start; i<=end; i++){
        arr[i] = temp[cur];
        cur++;
    }
}

void seqMergeSort(int *arr, int start, int end){
    if(start < end){
        int mid = (start + end) / 2;
        seqMergeSort(arr, start, mid);
        seqMergeSort(arr, mid+1, end);
        merge(arr, start, mid, end);
    }
}

void seqMerge(int *a){
    int arr[N];
    double start, end;
    for(int i=0;i<N;i++){
        arr[i] = a[i];
```

```
    }

    start = omp_get_wtime();
    seqMergeSort(arr, 0, N-1);
    end = omp_get_wtime();
    cout<<"The time taken for Serial Merge Sort : "<<(end-start)<<endl;
    cout<<endl;
}

void parMergeSort(int *arr, int start, int end){
    if(start < end){
        int mid = (start + end) / 2;

        #pragma omp parallel sections num_threads(2)
        {
            #pragma omp section
            {
                    parMergeSort(arr, start, mid);
            }
            #pragma omp section
            {
                    parMergeSort(arr, mid+1, end);
            }
        }
        merge(arr, start, mid, end);
    }
}

void parMerge(int *a){
    int arr[N];
    double start,end;
    for(int i=0; i<N; i++){
        arr[i] = a[i];
    }

    start = omp_get_wtime();
    parMergeSort(arr, 0, N-1);
    end = omp_get_wtime();
    cout<<"The time taken for Parallel Merge Sort : "<<(end-start)<<endl;
    cout<<endl;
}

int main(){
        for(int i=0;i<N;i++) {
                a[i]=rand()%N;
        }
        seqBubbleSort(a);
        parBubbleSort(a);
        seqMerge(a);
        parMerge(a);
}
```

OUPUT:


sumit@sumit-HP-Pavili
(base) sumit@sumit-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/HPC/Assign3$ g++ -fopenmp sort.cpp
(base) sumit@sumit-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/HPC/Assign3$ ./a.out
The time taken for Serial Bubble Sort : 10.7821
The time taken for Parallel Bubble Sort : 7.31315
The time taken for Serial Merge Sort : 0.163477
The time taken for Parallel Merge Sort : 0.0522328
(base) sumit@sumit-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/HPC/Assign3$