Assignment 2

Bubble Sort Implementation

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

using namespace std;

void sequentialBubbleSort(int *, int);
void parallelBubbleSort(int *, int);
void swap(int &, int &);

void sequentialBubbleSort(int *a, int n)
{
    int swapped;
    for (int i = 0; i < n; i++)
    {
        swapped = 0;
        for (int j = 0; j < n - 1; j++)
        {
            if (a[j] > a[j + 1])
            {
                swap(a[j], a[j + 1]);
                swapped = 1;
            }
        }

        if (!swapped)
        break;
    }
}

void parallelBubbleSort(int *a, int n)
{
    int swapped;
    for (int i = 0; i < n; i++)
    {
        swapped = 0;
        int first=i%2;
#pragma omp parallel for shared(a,first)
        for (int j = first; j < n - 1; j++)
        {
            if (a[j] > a[j + 1])
            {
                swap(a[j], a[j + 1]);
                swapped = 1;
            }
        }

        if (!swapped)
            break;
    }
}

void swap(int &a, int &b)
{
    int test;
    test = a;
    a = b;
    b = test;
}

int main()
{
    int *a, n;
    cout << "\n enter total no of elements=>";
    cin >> n;
    a = new int[n];
    cout << "\n enter elements=>";
    for (int i = 0; i < n; i++)
    {
        cin >> a[i];
    }

    double start_time = omp_get_wtime(); // start timer for sequential algorithm
    sequentialBubbleSort(a, n);
    double end_time = omp_get_wtime(); // end timer for sequential algorithm

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

    cout << "Time taken by sequential algorithm: " << end_time - start_time << " seconds" << endl;
    start_time = omp_get_wtime(); // start timer for parallel algorithm
    parallelBubbleSort(a, n);
    end_time = omp_get_wtime(); // end timer for parallel algorithm

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

    cout << "Time taken by parallel algorithm: " << end_time - start_time << " seconds" <<
endl;

    delete[] a; // Don't forget to free the allocated memory

    return 0;
}
```

OUTPUT:

```
enter total no of elements=>5
enter elements=>5 3 1 4 2

sorted array is=>1
2
3
4
5
Time taken by sequential algorithm: 0.000123 seconds
```

```
sorted array is=>1
2
3
4
5
Time taken by parallel algorithm: 0.000098 seconds
```

Merge Sort Implementation

```
#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;

      #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,j,k;
    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];
    }

    return 0;
}
```

OUTPUT:

```
enter total no of elements=>8
enter elements=>6 3 9 1 5 8 2 7

Sequential Time: 0.000456

Parallel Time: 0.000312

sorted array is=>
1
2
3
5
6
7
8
9
```