NAME : SUPRIYA KANSE (CO4029)

Experiment 2

Parallel Bubble Sort

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

void bubble(int *, int);
void swap(int &, int &);

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

int main(){
    int *a,n;
    cout<<"\n Enter total number: ";
    cin>>n;
    a=new int[n];
    cout<<" Enter number: ";
    for(int i=0;i<n;i++){
            cin>>a[i];
    }

    bubble(a,n);

    cout<<" Sorted array: ";
    for(int i=0;i<n;i++){
            cout<<a[i]<<" ";
```

```
    }
    return 0;
}
```

```
--- Parallel Bubble Sort ---
Enter total number: 6
Enter number: 10 17 15 12 9 11
Sorted array: 9 10 11 12 15 17
```

Parallel Merge Sort

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <thread>
using namespace std;

void merge(vector<int>& arr, int l, int m, int r) {
    int n1 = m - l + 1;
    int n2 = r - m;

    vector<int> L(n1), R(n2);

    for (int i = 0; i < n1; i++) {
        L[i] = arr[l + i];
    }
    for (int j = 0; j < n2; j++) {
        R[j] = arr[m + 1 + j];
    }

    int i = 0, j = 0, k = l;

    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        }
        else {
            arr[k] = R[j];
            j++;

    }
```

```
        k++;
    }

    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }}

void parallel_merge_sort(vector<int>& arr, int l, int r, int threads) {
    if (l < r) {
        int m = l + (r - l) / 2;

        if (threads > 1) {
            thread left(parallel_merge_sort, ref(arr), l, m, threads / 2);
            thread right(parallel_merge_sort, ref(arr), m + 1, r, threads / 2);

            left.join();
            right.join();
        }
        else {
            parallel_merge_sort(arr, l, m, 1);
            parallel_merge_sort(arr, m + 1, r, 1);
        }

        merge(arr, l, m, r);
    }}

int main() {
    int m;
    vector<int> arr;
    //= {10, 7, 8, 9, 1, 5};
    cout<<"\n--- Parallel Merge Sort ---";
    cout<<"\n Enter total number: ";
    cin>>m;

    cout<<" Enter number: ";
    int tmp;
    for(int i=0;i<m;i++){
            cin>>tmp;
            arr.push_back(tmp);
    }
```

```
    int n = arr.size();
    int threads = 4;

    parallel_merge_sort(arr, 0, n - 1, threads);

    cout<<" Sorted array: ";
    for (auto x: arr) {
        cout << x << " ";
    }
    cout << endl;

    return 0;
}
```