CODE:

```
#include <iostream>
#include <omp.h>
#include <vector>
#include <cstdlib>
#include <ctime>
using namespace std;
void bubbleSort(vector<int>& arr) {
    int n = arr.size();
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1])
                swap(arr[j], arr[j + 1]);
        }
    }
}
void parallelBubbleSort(vector<int>& arr) {
    int n = arr.size();
    for (int i = 0; i < n - 1; i++) {
        #pragma omp parallel for
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1])
                swap(arr[j], arr[j + 1]);
        }
    }
}
void merge(vector<int>& arr, int left, int mid, int right) {
    vector<int> temp(right - left + 1);
    int i = left, j = mid + 1, k = 0;
    while (i <= mid && j <= right) {
        if (arr[i] <= arr[j])
            temp[k++] = arr[i++];
        else
            temp[k++] = arr[j++];
    }
    while (i <= mid) temp[k++] = arr[i++];
    while (j <= right) temp[k++] = arr[j++];
    for (i = left, k = 0; i <= right; i++, k++) arr[i] = temp[k];
}
void mergeSort(vector<int>& arr, int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
        merge(arr, left, mid, right);
    }
}
void parallelMergeSort(vector<int>& arr, int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;
        #pragma omp parallel sections
        {
            #pragma omp section
            parallelMergeSort(arr, left, mid);
            #pragma omp section
            parallelMergeSort(arr, mid + 1, right);
        }
        merge(arr, left, mid, right);
    }
}
int main() {
    int n;
    cout << "Enter the number of elements: ";
    cin >> n;
    vector<int> arr(n), arr1, arr2, arr3, arr4;
    cout << "Enter the elements: ";
    for (int i = 0; i < n; i++) cin >> arr[i];
    arr1 = arr2 = arr3 = arr4 = arr;
    double start, end;
    start = omp_get_wtime();
    bubbleSort(arr1);
    end = omp_get_wtime();
    cout << "Sequential Bubble Sort Time: " << (end - start) << " seconds" << endl;
    start = omp_get_wtime();
    parallelBubbleSort(arr2);
    end = omp_get_wtime();
    cout << "Parallel Bubble Sort Time: " << (end - start) << " seconds" << endl;
    start = omp_get_wtime();
    mergeSort(arr3, 0, n - 1);
    end = omp_get_wtime();
    cout << "Sequential Merge Sort Time: " << (end - start) << " seconds" << endl;
    start = omp_get_wtime();
    parallelMergeSort(arr4, 0, n - 1);
    end = omp_get_wtime();
    cout << "Parallel Merge Sort Time: " << (end - start) << " seconds" << endl;
    return 0;
}
```

INPUT:

```
Enter the number of elements: 5
Enter the elements: 12 7 9 5 10
```

OUTPUT:

```
Sequential Bubble Sort Time: 0.000012 seconds
Parallel Bubble Sort Time: 0.000008 seconds
Sequential Merge Sort Time: 0.000010 seconds
Parallel Merge Sort Time: 0.000006 seconds
```