```
#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;
        #pragma omp parallel for shared(a, swapped) // added shared clause
        for (int j = 0; j < n - 1; j++) // start from 0 for every iteration
        {
            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: 0.000197296 seconds

sorted array is=>1
2
3
4
5
Time taken by parallel algorithm: 0.000133686 seconds
```