Practical 2 : Bubble Sort

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

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

int main()
{
    int *a, *b, n;
    cout << "\nEnter total no of elements : ";
    cin >> n;
    a = new int[n];
    b = new int[n];
    for (int i = 0; i < n; i++)
    {
        int random_value = rand()%1000000;
        a[i] = random_value;
        b[i] = random_value;
    }

    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
    double seq_time = end_time - start_time;

    cout << "\nSorted array ( first 20 elements ) : \n";
    for (int i = 0; i < 20; i++)
    {
        cout << a[i] <<", ";
    }

    start_time = omp_get_wtime(); // start timer for parallel algorithm
    parallelBubbleSort(b, n);
    end_time = omp_get_wtime(); // end timer for parallel algorithm
    double parallel_time = end_time - start_time;

    cout << "\n\nSorted array ( first 20 elements using parallel algorithm ) : \n";
    for (int i = 0; i < 20; i++)
    {
```

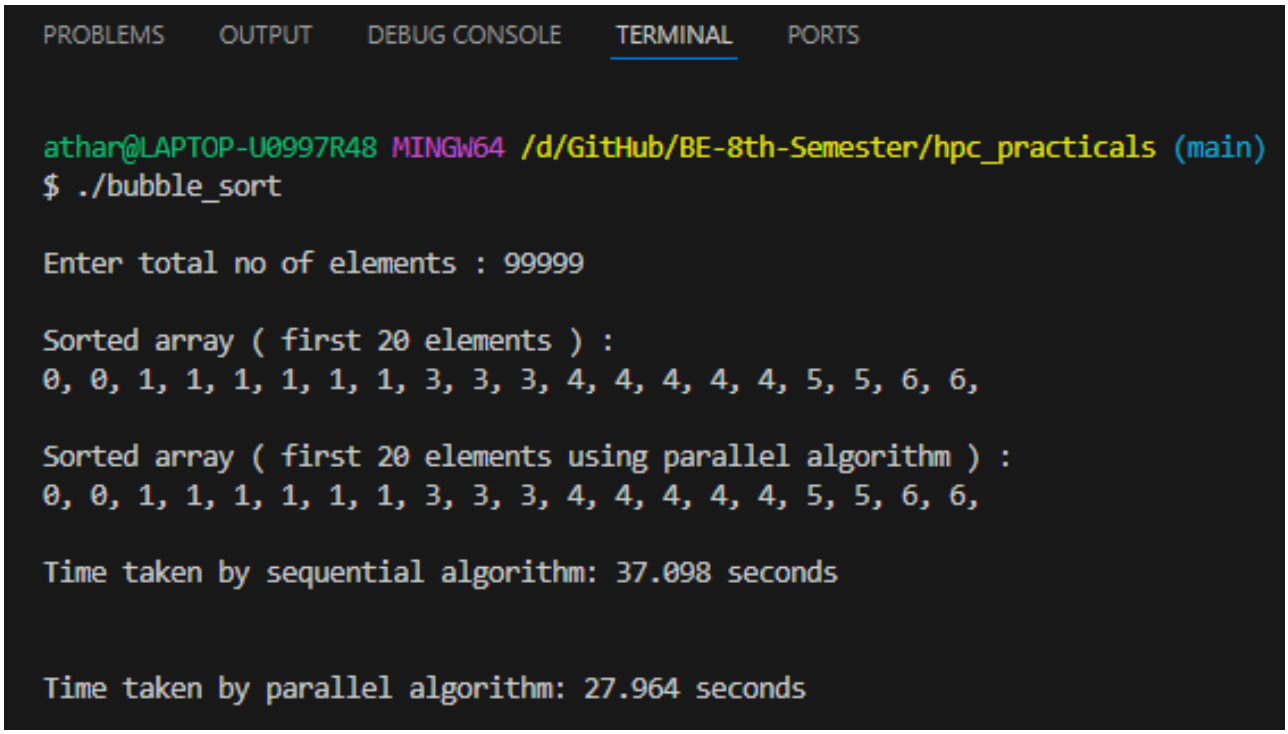
```
        cout << b[i] <<", ";
    }

    cout << "\n\nTime taken by sequential algorithm: " << seq_time << " seconds" << endl;
    cout << "\n\nTime taken by parallel algorithm: " << parallel_time << " seconds" << endl;

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

    return 0;
}
```

Output :



Total number of elements : 99999
Time taken by Sequential Algo : 37.098 seconds
Time taken by Parallel Algo : 27.964 seconds

```
athar@LAPTOP-U0997R48 MINGW64 /d/GitHub/BE-8th-Semester/hpc_practicals/Codes (main)
$ g++ -fopenmp bubble_sort.cpp -o bubble_sort

athar@LAPTOP-U0997R48 MINGW64 /d/GitHub/BE-8th-Semester/hpc_practicals/Codes (main)
$ ./bubble_sort

Enter total no of elements : 130867

Sorted array ( first 20 elements ) :
0, 0, 1, 1, 1, 1, 1, 1, 1, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4,

Sorted array ( first 20 elements using parallel algorithm ) :
0, 0, 1, 1, 1, 1, 1, 1, 1, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4,

Time taken by sequential algorithm: 121.994 seconds


Time taken by parallel algorithm: 61.051 seconds
```

Total number of elements : 130867
Time taken by Sequential Algo : 121.994 seconds
Time taken by Parallel Algo : 61.051 seconds