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

using namespace std;

void bubble(int array[], int n){
    for (int i = 0; i < n - 1; i++){
        for (int j = 0; j < n - i - 1; j++){
            if (array[j] > array[j + 1]) swap(array[j], array[j + 1]);
        }
    }
}

void pBubble(int array[], int n){
    //Sort odd indexed numbers
    for(int i = 0; i < n; ++i){
        #pragma omp for
        for (int j = 1; j < n; j += 2){
        if (array[j] < array[j-1])
        {
          swap(array[j], array[j - 1]);
        }
    }

    // Synchronize
    #pragma omp barrier

    //Sort even indexed numbers
    #pragma omp for
    for (int j = 2; j < n; j += 2){
      if (array[j] < array[j-1])
      {
        swap(array[j], array[j - 1]);
      }
    }
  }
}

void printArray(int arr[], int n){
    for(int i = 0; i < n; i++) cout << arr[i] << " ";
    cout << "\n";
}

int main(){
    // Set up variables
    int n = 10;
    int arr[n];
    int brr[n];
    double start_time, end_time;

    // Create an array with numbers starting from n to 1
    for(int i = 0, j = n; i < n; i++, j--) arr[i] = j;

    // Sequential time
    start_time = omp_get_wtime();
    bubble(arr, n);
    end_time = omp_get_wtime();
```

```
    cout << "Sequential Bubble Sort took : " << end_time - start_time <<
" seconds.\n";
    printArray(arr, n);

    // Reset the array
    for(int i = 0, j = n; i < n; i++, j--) arr[i] = j;

    // Parallel time
    start_time = omp_get_wtime();
    pBubble(arr, n);
    end_time = omp_get_wtime();
    cout << "Parallel Bubble Sort took : " << end_time - start_time << "
seconds.\n";
    printArray(arr, n);
}
```