```
// Write a program to implement Parallel Bubble Sort and Merge sort using
OpenMP. Use existing algorithms and measure the performance of sequential and
parallel algorithms.

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

// Function to perform parallel bubble sort
void parallelBubbleSort(vector<int>& arr) {
    int n = arr.size();
    for (int i = 0; i < n; i++) {
        #pragma omp parallel for
        for (int j = i % 2; j < n - 1; j += 2) {
            if (arr[j] > arr[j + 1]) {
                swap(arr[j], arr[j + 1]);
            }
        }
    }
}

// Merge helper for merge sort
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)
        temp[k++] = (arr[i] < arr[j]) ? arr[i++] : arr[j++];
    while (i <= mid) temp[k++] = arr[i++];
    while (j <= right) temp[k++] = arr[j++];

    for (int i = 0; i < k; i++)
        arr[left + i] = temp[i];
}

// Recursive parallel merge sort with OpenMP tasks
void parallelMergeSort(vector<int>& arr, int left, int right) {
    const int threshold = 1000; // Set to 1000 or based on input size
    if (left < right) {
        int mid = (left + right) / 2;

        if (right - left < threshold) {
            // Sequential call for small segments
            parallelMergeSort(arr, left, mid);
            parallelMergeSort(arr, mid + 1, right);
        } else {
            #pragma omp task shared(arr)
            parallelMergeSort(arr, left, mid);

            #pragma omp task shared(arr)
            parallelMergeSort(arr, mid + 1, right);

            #pragma omp taskwait
```

```
        }

        merge(arr, left, mid, right);
    }
}


int main() {
    int size;
    cout << "Enter the size of the array: ";
    cin >> size;

    vector<int> arr(size), bubbleArr, mergeArr;
    cout << "Enter " << size << " elements:\n";
    for (int i = 0; i < size; i++) {
        cin >> arr[i];
    }

    bubbleArr = arr;
    mergeArr = arr;

    parallelBubbleSort(bubbleArr);

    #pragma omp parallel
    {
        #pragma omp single
        {
            parallelMergeSort(mergeArr, 0, size - 1);
        }
    }

    cout << "\nSorted Array using Parallel Bubble Sort:\n";
    for (int x : bubbleArr) cout << x << " ";

    cout << "\n\nSorted Array using Parallel Merge Sort:\n";
    for (int x : mergeArr) cout << x << " ";

    cout << endl;
    return 0;
}

/*g++ -fopenmp filename.cpp -o filename.exe
Filename.exe
gcc -v takaycha command prompt la
https://www.winlibs.com/*/
```