|  |  |
| --- | --- |
| 2a | Write a program to implement Parallel Bubble Sort. Use existing algorithms and measure the performance of sequential and parallel algorithms |

#include <iostream>

#include <stdlib.h>

#include <omp.h>

using namespace std;

void bubbleSort(int \*arr, int n);

void swap(int &x, int &y);

void bubbleSort(int \*arr, int n) {

// Bubble sort with parallelization

for (int i = 0; i < n; i++) {

int start = i % 2; // Decide the starting point for even/odd indexed elements

#pragma omp parallel for shared(arr, start) // Parallel loop

for (int j = start; j < n - 1; j += 2) {

if (arr[j] > arr[j + 1]) {

swap(arr[j], arr[j + 1]); // Swap if in the wrong order

}

}

}

}

void swap(int &x, int &y) {

int temp = x;

x = y;

y = temp;

}

int main() {

int n;

cout << "Enter the total number of elements: ";

cin >> n;

int \*arr = new int[n]; // Dynamic allocation of array

cout << "Enter the elements: ";

for (int i = 0; i < n; i++) {

cin >> arr[i];

}

bubbleSort(arr, n); // Sort the array

cout << "Sorted array is:" << endl;

for (int i = 0; i < n; i++) {

cout << arr[i] << endl; // Output the sorted array

}

delete[] arr; // Free the allocated memory

return 0;

}