**Code:**

#include<iostream>

#include<bits/stdc++.h>

using namespace std;

void merge(int arr[], int p, int q, int r) {

int n1 = q - p + 1;

int n2 = r - q;

int L[n1], M[n2];

for (int i = 0; i < n1; i++)

L[i] = arr[p + i];

for (int j = 0; j < n2; j++)

M[j] = arr[q + 1 + j];

int i, j, k;

i = 0;

j = 0;

k = p;

while (i < n1 && j < n2) {

if (L[i] <= M[j]) {

arr[k] = L[i];

i++;

}

else {

arr[k] = M[j];

j++;

}

k++;

}

while (i < n1) {

arr[k] = L[i];

i++;

k++;

}

while (j < n2) {

arr[k] = M[j];

j++;

k++;

}

}

void mergeSort(int arr[], int l, int r) {

if (l < r) {

int m = l + (r - l) / 2;

mergeSort(arr, l, m);

mergeSort(arr, m + 1, r);

merge(arr, l, m, r);

}

}

void merge\_para(int arr[],int l,int r){

if(l<r){

int m=l+(r-l)/2;

#pragma omp parallel sections

{

#pragma omp section

{

merge\_para(arr,l,m);

}

#pragma omp section

{

merge\_para(arr,m+1,r);

}

}

merge(arr,l,m,r);

}

}

void bubbleSort(int arr[], int n)

{

int i, j;

for (i = 0; i < n - 1; i++){

for (j = 0; j < n - i - 1; j++){

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

swap(arr[j], arr[j + 1]);

}

}

}

}

void swap\_para(int \*a, int \*b){

int temp=\*a;

\*a=\*b;

\*b=temp;

}

void bubble\_para(int arr[], int n){

int i=0, j=0;

int f;

for (i = 0; i < n - 1; i++){

f=i%2;

#pragma omp parallel for default(none), shared(arr,first,n)

for (j = f; j < n - 1; j++){

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

swap\_para(&arr[j], &arr[j + 1]);

}

}

}

}

void printArray(int arr[], int size) {

for (int i = 0; i < size; i++)

cout << arr[i] << " ";

cout << endl;

}

int main() {

int n;

cout<<"Enter no of elements in array:";

cin>>n;

int arr1[n],arr2[n],arr3[n],arr4[n];

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

//cin>>arr[i];

arr1[i]=rand()%n;

arr2[i]=arr1[i];

arr3[i]=arr1[i];

arr4[i]=arr1[i];

}

printArray(arr1,n);

auto start = chrono :: steady\_clock :: now();

mergeSort(arr1, 0, n - 1);

auto end = chrono :: steady\_clock :: now();

cout << " Merge Sorted array: \n";

printArray(arr1, n);

chrono::duration<double,micro>fp=end-start;

cout<<fp.count()<<" microseconds"<<endl;

auto start1 = chrono :: steady\_clock :: now();

merge\_para(arr2,0,n-1);

auto end1 = chrono :: steady\_clock :: now();

cout << "Parallel Merge Sorted array: \n";

printArray(arr2, n);

chrono::duration<double,micro>fp1=end1-start1;

cout<<fp1.count()<<" microseconds"<<endl;

auto start2 = chrono :: steady\_clock :: now();

bubbleSort(arr3, n);

auto end2 = chrono :: steady\_clock :: now();

cout << "Bubble Sorted array: \n";

printArray(arr3, n);

chrono::duration<double,micro>fp2=end2-start2;

cout<<fp2.count()<<" microseconds"<<endl;

auto start3 = chrono::steady\_clock::now();

bubble\_para(arr4,n);

auto end3 = chrono::steady\_clock::now();

cout << "Parallel Bubble Sorted array: \n";

printArray(arr4, n);

chrono::duration<double,micro>ft3=end3-start3;

cout<<ft3.count()<<" microseconds"<<endl;

return 0;

}

**Output:**

