#include<iostream>

#include<stdlib.h>

#include<omp.h>

using namespace std;

void mergesort(int a[],int i,int j);

void merge(int a[],int i1,int j1,int i2,int j2);

void mergesort(int a[],int i,int j)

{

int mid;

if(i<j)

{

mid=(i+j)/2;

#pragma omp parallel sections

{

#pragma omp section

{

mergesort(a,i,mid);

}

#pragma omp section

{

mergesort(a,mid+1,j);

}

}

merge(a,i,mid,mid+1,j);

}

}

void merge(int a[],int i1,int j1,int i2,int j2)

{

int temp[1000];

int i,j,k;

i=i1;

j=i2;

k=0;

while(i<=j1 && j<=j2)

{

if(a[i]<a[j])

{

temp[k++]=a[i++];

}

else

{

temp[k++]=a[j++];

}

}

while(i<=j1)

{

temp[k++]=a[i++];

}

while(j<=j2)

{

temp[k++]=a[j++];

}

for(i=i1,j=0;i<=j2;i++,j++)

{

a[i]=temp[j];

}

}

int main()

{

int \*a,n,i;

cout<<"\n enter total no of elements=>";

cin>>n;

a= new int[n];

cout<<"\n enter elements=>";

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

{

cin>>a[i];

}

// start=.......

//#pragma omp…..

mergesort(a, 0, n-1);

// stop…….

cout<<"\n sorted array is=>";

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

{

cout<<"\n"<<a[i];

}

// Cout<<Stop-Start

return 0;

}

Second Code:

#include <iostream>

#include <omp.h>

void merge(int\* arr, int l, int m, int r) {

int i, j, k;

int n1 = m - l + 1;

int n2 = r - m;

int L[n1], R[n2];

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

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

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

R[j] = arr[m + 1 + j];

i = 0;

j = 0;

k = l;

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

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

arr[k] = L[i];

i++;

}

else {

arr[k] = R[j];

j++;

}

k++;

}

while (i < n1) {

arr[k] = L[i];

i++;

k++;

}

while (j < n2) {

arr[k] = R[j];

j++;

k++;

}

}

void mergeSort(int\* arr, int l, int r) {

if (l < r) {

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

#pragma omp parallel sections

{

#pragma omp section

{

mergeSort(arr, l, m);

}

#pragma omp section

{

mergeSort(arr, m + 1, r);

}

}

merge(arr, l, m, r);

}

}

int main() {

int arr[] = { 12, 11, 13, 5, 6, 7 };

int n = sizeof(arr) / sizeof(arr[0]);

double start, stop;

std::cout << "Given array is: ";

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

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

std::cout << std::endl;

start = omp\_get\_wtime();

#pragma omp parallel

{

mergeSort(arr, 0, n - 1);

}

stop = omp\_get\_wtime();

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

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

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

std::cout << std::endl;

std::cout<<stop-start;

return 0;

}

OUTPUT:

