#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;

double start\_time, end\_time, seq\_time, par\_time;

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];

}

// Sequential algorithm

start\_time = omp\_get\_wtime();

mergesort(a, 0, n-1);

end\_time = omp\_get\_wtime();

seq\_time = end\_time - start\_time;

cout << "\nSequential Time: " << seq\_time << endl;

// Parallel algorithm

start\_time = omp\_get\_wtime();

#pragma omp parallel

{

#pragma omp single

{

mergesort(a, 0, n-1);

}

}

end\_time = omp\_get\_wtime();

par\_time = end\_time - start\_time;

cout << "\nParallel Time: " << par\_time << endl;

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

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

{

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

}

return 0;

}