/\*

Title:Implement a parallel quick sort algorithm.

Class:TE B

Roll no:63

\*/

#include<stdio.h>

#include<omp.h>

int partition( int a[],int low,int high)

{

int i,j,pivot,tmp;

if(low<high)

{

i=low;

j=high;

pivot=i;

while(i<j)

{

while(a[pivot]>=a[i] && i<high)

{

i++;

}

while(a[pivot]<a[j])

{

j--;

}

if(i<j)

{

tmp=a[i];

a[i]=a[j];

a[j]=tmp;

}

}

tmp=a[j];

a[j]=a[pivot];

a[pivot]=tmp;

}

return(j);

}

void quicksort(int a[],int low,int high)

{

if(low<high)

{

int p = partition(a,low,high);

printf("\n Partition occured at %d",p);

#pragma omp parallel sections

{

#pragma omp section

{

quicksort(a,low,p-1);

}

#pragma omp section

{

quicksort(a,p+1,high);

}

}

}

}

int main()

{

int i,n,a[10];

double start\_time,total\_time;

printf("Enter number of elements:");

scanf("%d",&n);

printf("\n Enter the elememnts of array:");

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

scanf("%d",&a[i]);

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

quicksort(a,0,n-1);

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

printf("\n Sorted array is:");

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

printf("%d\t",a[i]);

printf("\n");

printf("\n Start time:%f",start\_time);

printf("\n End time:%f",total\_time);

printf("\n");

return 0;

}

/\*Output:

unix@unix-HP-280-G1-MT:~/Desktop$ gcc -fopenmp quicksort1.c

unix@unix-HP-280-G1-MT:~/Desktop$ ./a.out

Enter number of elements:7

Enter the elememnts of array:10

5

9

4

12

6

15

Partition occured at 4

Partition occured at 2

Partition occured at 0

Partition occured at 5

Sorted array is:4 5 6 9 10 12 15

Start time:3549.653413

End time:0.002718

\*/