**ASSIGNMENT NO.**

NAME: SHRUTI DILIP BHUJANGE

CLASS: BE COMP-1 ROLL NO.: 402006

PROGRAM:

#include<stdio.h>

#include<omp.h>

int part(int arr[],int low,int high)

{

int pvt,i,j,temp,t;

pvt=arr[high];

i=low-1;

#pragma omp parallel for

for(j=low;j<high;j++)

{

if(arr[j] <= pvt)

{

i=i+1;

temp=arr[j];

arr[j]=arr[i];

arr[i]=temp;

}

}

t=arr[high];

arr[high]=arr[i+1];

arr[i+1]=t;

return i+1;

}

void quick(int arr[],int low,int high)

{

int p,tid,tid2;

if (low < high)

{

p=part(arr,low,high);

#pragma omp parallel sections

{

#pragma omp section

{

tid = omp\_get\_thread\_num();

printf("Thread %d working on index %d to %d.\n",tid,low,p-1);

quick(arr,low,p-1);

}

#pragma omp section

{

tid2 = omp\_get\_thread\_num();

printf("Thread %d working on index %d to %d.\n",tid2,p+1,high);

quick(arr,p+1,high);

}

}

}

}

int main()

{

int n,i,j;

printf("Enter number of elements:");

scanf("%d",&n);

printf("Enter elements of array:\n");

int arr[n];

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

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

quick(arr,0,n-1);

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

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

printf("%d\t",arr[j]);

printf("\n");

return 0;

}