**Task 4: D&C – Merge Sort**

Implement a parallelized Merge Sort algorithm to sort a given set of elements and determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted and plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.

#include <stdio.h>

#include <stdlib.h>

#include<sys/time.h>

#include<omp.h>

Void simple merge(int a[],int low,int mid,int high)

{

int i,j,k,c[10000];

i=low;

j=mid+1;k=low;

int tid;

omp\_set\_num\_threads(5);

{

#pragma omp paralleltid=omp\_get\_thread\_nu();#pragma omp while(i<=mid&&j<=high)

{

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

{

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

k++;

}

else

{

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

k++;

}

}

}

while(i<=mid)

{

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

k++;

}

while(j<=high)

{

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

k++;

}

for(k=low;k<=high;k++)a[k]=c[k];

}

voidmerge(inta[],intlow,inthigh)

{

int mid;if(low<high)

{

mid=(low+high)/2;merge(a,low,mid);merge(a,mid+1,high);simplemerge(a,low,mid,high);

}

}

voidgetnumber(inta[],intn)

{

int i;for(i=0;i<n;i++)a[i]=rand()%10000;

}

Int main()

{

FILE\*fp;

int a[10000],i;structtimevaltv;

double start,end,elapse;fp=fopen("m.txt","w");printf("Num.time");for(i=0;i<9999;i+=100)

{

getnumber(a,i);gettimeofday(&tv,NULL);start=tv.tv\_sec+(tv.tv\_usec/100000.0);merge(a,0,i-1);gettimeofday(&tv,NULL);end=tv.tv\_sec+(tv.tv\_usec/100000.0);elapse=end-start;

if(elapse>=0)

fprintf(fp,"%d\t%lf\n",i,elapse);

}

fclose(fp);system("gnuplot");return0; }

RESULT:

Thus, successfully completed Merge Sort and also executed the output.