```
 #include <stdio.h>
 #include <stdlib.h>
 #include <sys/time.h>
 #include <omp.h>


/
**********************************************************************
**********

*Function   : simplemerge
*Description    : Function to merge two sorted arrays
*Input parameters:
*   int a[] - iaArray to hold integers
*   int low - start index of the subiaArray to be sorted
*   int mid - mid index of the subiaArray to be sorted
*   int right   - end index of the subiaArray to be sorted
*RETURNS    : no value

**********************************************************************
**********/

void simplemerge(int a[], int low, int mid, int high)
{
    int i,j,k,c[20000];
    i=low;
    j=mid+1;
    k=low;
    int tid;
    omp_set_num_threads(10);
    {
        tid=omp_get_thread_num();
        while(i<=mid&&j<=high)
        {
            if(a[i] < a[j])
            {
                c[k]=a[i];
                //printf("%d%d",tid,c[k]);
                i++;
                k++;
            }
            else
            {
                c[k]=a[j];
                //printf("%d%d", tid, c[k]);
                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];
}

/
**********************************************************************
**********
*Function   : merge
*Description    : Function to sort elements in an iaArray using
Quick Sort
*Input parameters:
    int a[] - iaArray to hold integers
    int low - start index of the array to be sorted
    int high- end index of the array to be sorted
*RETURNS    : no value
**********************************************************************
**********/

void merge(int a[],int low,int high)
{
    int mid;
    if(low < high)
    {
        mid=(low+high)/2;
        merge(a,low,mid);
        merge(a,mid+1,high);
        simplemerge(a,low,mid,high);
    }
}

void getnumber(int a[], int n)
{
    int i;
    for(i=0;i < n;i++)
        a[i]=rand()%100;
}

/
**********************************************************************
**********
*Function   : main
*Input parameters: no
*RETURNS    :   0 on success
**********************************************************************
**********/
int main()
```

```
{
    FILE *fp;
    int a[2000],i;
    struct timeval tv;
    double start, end, elapse;
    fp=fopen("mergesort.txt","w");
    for(i=10;i<=1000;i+=10)
    {
        getnumber(a,i);
        gettimeofday(&tv,NULL);
        start=tv.tv_sec+(tv.tv_usec/1000000.0);
        merge(a,0,i-1);
        gettimeofday(&tv,NULL);
        end=tv.tv_sec+(tv.tv_usec/1000000.0);
        elapse=end-start;
        fprintf(fp,"%d\t%lf\n",i,elapse);
    }
    fclose(fp);
    system("gnuplot");
    return 0;
}
```

mergesort.gpl

Gnuplot script file for plotting data in file "mergesort.txt" This file is called mergesort.gpl

```
set terminal png font arial
set title "Time Complexity for Merge Sort"
set autoscale
set xlabel "Size of Input"
set ylabel "Sorting Time (microseconds)"
set grid
set output "mergesort.png"
plot "mergesort.txt" t "Merge Sort" with lines
```