Q2.write a program to sort a given set of elements using the Merge sort method 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 t elements can be read from a file or can be generated using the random number generator.

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
void merge(int arr[], int low, int mid, int high){
int n1 = mid - low + 1;
int n2 = high - mid;
int L[n1], R[n2];
 for (int i = 0; i < n1; i++)
L[i] = arr[low+i];
 for (int j = 0; j < n2; j++)
R[j] = arr[mid+j+1];

int i= 0;
int j = 0;
int k = low;
while (i if(L[i] <= R[j]){
 arr[k] = L[i];
 i++;
}
else{
```

```
 arr[k] = R[j];
 j++;
}
k++;
}

while (i< n1){
 arr[k] = L[i];
 i++;
 k++;
}

while(j < n2){
  arr[k] = R[j];
  j++;
  k++;
 }
}

void mergesort(int arr[], int low, int high){
if (low < high){
int mid = low + (high-low)/2;
mergesort(arr, low, mid);
mergesort(arr, mid+1, high);
```

```
 merge(arr, low, mid, high);
 }
}

void display(int arr[], int n){
for (int i = 0; i < n; i++)
{
 printf("%d\t", arr[i]);
 }
}

int main(){
int n;
clock_t start, end  ;
double totalCPUtime;
printf("Enter the number of elemnts in the array  ");
scanf("%d", &n);
int arr[n];

for (int i =0;i arr[i] = (rand() % 101);
 printf("%d\t", arr[i]);
}
int low = 0;
int high = n-1;
```

```
printf("\nsorted array ..........\n");
start = clock();
mergesort(arr, low, high);
end = clock();
display(arr, n);
printf("\n\n");

totalCPUtime = ((double)(end - start));
printf("\ntotal CPU time in ms: %f", totalCPUtime);
totalCPUtime = ((double)(end - start)/CLOCKS_PER_SEC);
printf("\ntotal CPU time in s  %f", totalCPUtime);

return 0;
}
```