LAB -8
LAKSHMI S KUMAR

```
#include<stdlib.h>
 #include<stdio.h>
 #include<time.h>
 void merge(int arr[], int l, int m, int r)
 {
    int i, j, k;
    int n1 = m - l + 1;
    int n2 =  r - m;


    int L[n1], R[n2];


    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1+ j];
    i = 0;
    j = 0;
    k = l;
    while (i < n1 && j < n2)
    {
        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 l, int r)
{
    if (l < r)
    {
        int m = l+(r-l)/2;
        mergeSort(arr, l, m);
        mergeSort(arr, m+1, r);

        merge(arr, l, m, r);
```

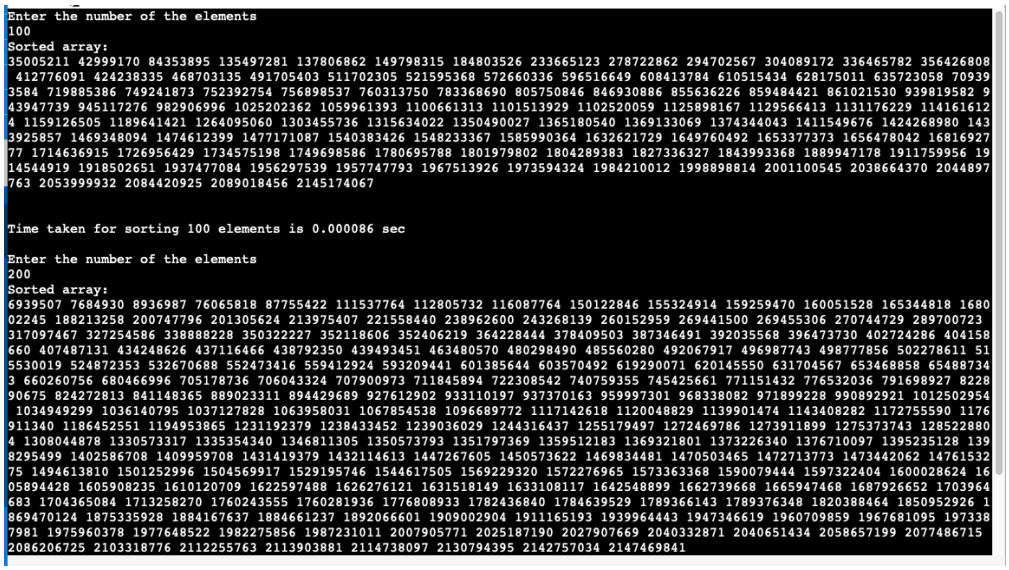
```
   }
}


void printArray(int A[], int size)
{
   int i;
   for (i=0; i < size; i++)
       printf("%d ", A[i]);
   printf("\n");
}


int main()
{
int i,n,sort;
clock_t start,end;
while(1)
{
printf("Enter the number of the elements\n");
scanf("%d",&n);
if(n==-1)
   break;
int a[n];
for(i=0;i<n;i++)
{
a[i]=rand();
}


start=clock();
mergeSort(a, 0, n - 1);
printf("Sorted array:\n");
```

```
    printArray(a, n);
 end=clock();
 double time_taken=((double)end-start)/CLOCKS_PER_SEC;
 printf("\n\n");
 printf("Time taken for sorting %d elements is %f
 sec\n",n,time_taken);
 printf("\n");
 }
 }
```



```
Enter the number of the elements
100
Sorted array:
35005211 42999170 84353895 135497281 137806862 149798315 184803526 233665123 278722862 294702567 304089172 336465782 356426808
 412776091 424238335 468703135 491705403 511702305 521595368 572660336 596516649 608413784 610515434 628175011 635723058 70939
3584 719885386 749241873 752392754 756898537 760313750 783368690 805750846 846930886 855636226 859484421 861021530 939819582 9
43947739 945117276 982906996 1025202362 1059961393 1100661313 1101513929 1102520059 1125898167 1129566413 1131176229 114161612
4 1159126505 1189641421 1264095060 1303455736 1315634022 1350490027 1365180540 1369133069 1374344043 1411549676 1424268980 143
3925857 1469348094 1474612399 1477171087 1540383426 1548233367 1585990364 1632621729 1649760492 1653377373 1656478042 16816927
77 1714636915 1726956429 1734575198 1749698586 1780695788 1801979802 1804289383 1827336327 1843993368 1889947178 1911759956 19
14544919 1918502651 1937477084 1956297539 1957747793 1967513926 1973594324 1984210012 1998898814 2001100545 2038664370 2044897
763 2053999932 2084420925 2089018456 2145174067


Time taken for sorting 100 elements is 0.000086 sec

Enter the number of the elements
200
Sorted array:
6939507 7684930 8936987 76065818 87755422 111537764 112805732 116087764 150122846 155324914 159259470 160051528 165344818 1680
02245 188213258 200747796 201305624 213975407 221558440 238962600 243268139 260152959 269441500 269455306 270744729 289700723
317097467 327254586 338888228 350322227 352118606 352406219 364228444 378409503 387346491 392035568 396473730 402724286 404158
660 407487131 434248626 437116466 438792350 439493451 463480570 480298490 485560280 492067917 496987743 498777856 502278611 51
5530019 524872353 532670688 552473416 559412924 593209441 601385644 603570492 619290071 620145550 631704567 653468858 65488734
3 660260756 680466996 705178736 706043324 707900973 711845894 722308542 740759355 745425661 771151432 776532036 791698927 8228
90675 824272813 841148365 889023311 894429689 927612902 933110197 937370163 959997301 968338082 971899228 990892921 1012502954
 1034949299 1036140795 1037127828 1063958031 1067854538 1096689772 1117142618 1120048829 1139901474 1143408282 1172755590 1176
911340 1186452551 1194953865 1231192379 1238433452 1239036029 1244316437 1255179497 1272469786 1273911899 1275373743 128522880
4 1308044878 1330573317 1335354340 1346811305 1350573793 1351797369 1359512183 1369321801 1373226340 1376710097 1395235128 139
8295499 1402586708 1409959708 1431419379 1432114613 1447267605 1450573622 1469834481 1470503465 1472713773 1473442062 14761532
75 1494613810 1501252996 1504569917 1529195746 1544617505 1569229320 1572276965 1573363368 1590079444 1597322404 1600028624 16
05894428 1605908235 1610120709 1622597488 1626276121 1631518149 1633108117 1642548899 1662739668 1665947468 1687926652 1703964
683 1704365084 1713258270 1760243555 1760281936 1776808933 1782436840 1784639529 1789366143 1789376348 1820388464 1850952926 1
869470124 1875335928 1884167637 1884661237 1892066601 1909002904 1911165193 1939964443 1947346619 1960709859 1967681095 197338
7981 1975960378 1977648522 1982275856 1987231011 2007905771 2025187190 2027907669 2040332871 2040651434 2058657199 2077486715
2086206725 2103318776 2112255763 2113903881 2114738097 2130794395 2142757034 2147469841
```

```
#include<stdio.
h>
                    #include<time.h>
                    #include <stdlib.h>
                    void swap(int* a, int* b)
                    {
                       int t = *a;
                       *a = *b;
                       *b = t;
                    }


                    int partition (int arr[], int low, int high)
```

```
{
   int pivot = arr[high];
   int i = (low - 1);
   for (int j = low; j <= high - 1; j++)
   {
       if (arr[j] < pivot)
       {
           i++;
           swap(&arr[i], &arr[j]);
       }
   }
   swap(&arr[i + 1], &arr[high]);
   return (i + 1);
}


void quickSort(int arr[], int low, int high)
{
   if (low < high)
   {
       int pi = partition(arr, low, high);
       quickSort(arr, low, pi - 1);
       quickSort(arr, pi + 1, high);
   }
}


void printArray(int arr[], int size)
{
   int i;
   for (i = 0; i < size; i++)
       printf("%d ",arr[i]);
   printf("\n");
```

```
}



int main()
{
int i,n,sort;
clock_t start,end;
while(1)
{
printf("Enter the number of the elements\n");
scanf("%d",&n);
if(n==-1)
   break;
int a[n];
for(i=0;i<n;i++)
{
a[i]=rand();
}
start=clock();
quickSort(a, 0, n - 1);
printf("Sorted array:\n");
   printArray(a, n);
end=clock();
double time_taken=((double)end-start)/CLOCKS_PER_SEC;
printf("\n\n");
printf("Time taken for sorting %d elements is %f
sec\n",n,time_taken);
printf("\n");
}
}
```

```
Enter the number of the elements
100
Sorted array:
35005211 42999170 84353895 135497281 137806862 149798315 184803526 233665123 278722862 294702567 304089172 336465782 356426808
 412776091 424238335 468703135 491705403 511702305 521595368 572660336 596516649 608413784 610515434 628175011 635723058 70939
3584 719885386 749241873 752392754 756898537 760313750 783368690 805750846 846930886 855636226 859484421 861021530 939819582 9
43947739 945117276 982906996 1025202362 1059961393 1100661313 1101513929 1102520059 1125898167 1129566413 1131176229 114161612
4 1159126505 1189641421 1264095060 1303455736 1315634022 1350490027 1365180540 1369133069 1374344043 1411549676 1424268980 143
3925857 1469348094 1474612399 1477171087 1540383426 1548233367 1585990364 1632621729 1649760492 1653377373 1656478042 16816927
77 1714636915 1726956429 1734575198 1749698586 1780695788 1801979802 1804289383 1827336327 1843993368 1889947178 1911759956 19
14544919 1918502651 1937477084 1956297539 1957747793 1967513926 1973594324 1984210012 1998898814 2001100545 2038664370 2044897
763 2053999932 2084420925 2089018456 2145174067


Time taken for sorting 100 elements is 0.000078 sec

Enter the number of the elements
```