PRACTICAL – 3

Write user defined functions for the following sorting methods and compare their performance by time measurement with random data and Sorted data.

1. Selection Sort
2. Bubble Sort
3. Insertion Sort
4. Merge Sort
5. Quick Sort

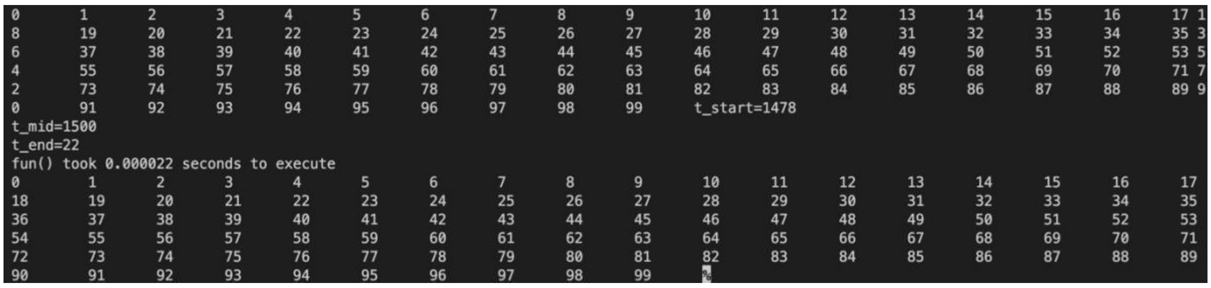
CODE:

```
#include<stdio.h>
#include<time.h>
#include <stdlib.h>


void selection(int arr[], int n) ;
void printArr(int arr[], int n);

int main(){     int
n=100;
    int arr[n];

    for(int i=0;i<n;i++){
arr[i]=rand();
    }

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

    clock_t t_start,t_end,t_mid;      t_start = clock();
//starting time after initializing data
printf("t_start=%lu\n",t_start);

    selection( arr, n);
          t_mid=clock();                                    //mid time after function
call     printf("t_mid=%lu\n",t_mid);

    t_end = t_mid - t_start;                              //ending time after mid - start
printf("t_end=%lu\n",t_end);
```

```
    double time_taken = ((double)t_end)/CLOCKS_PER_SEC;           //clock per sec is a macro
and 1cps= 1 million micro sec
    printf("fun() took %f seconds to execute \n", time_taken);

    printArr(arr, n);

    return 0;

}


void selection(int arr[], int n)
{       int i, j,
small;

    for (i = 0; i < n-1; i++)    // One by one move boundary of unsorted subarray
    {
        small = i; //minimum element in unsorted array
                    for (j = i+1; j <
n; j++)
if (arr[j] < arr[small])
            small = j;
// Swap the minimum element with the first element
int temp = arr[small];       arr[small] = arr[i];
    arr[i] = temp;
    }
}

void printArr(int arr[], int n) /* function to print the array */
{       int
i;
    for (i = 0; i < n; i++)
printf("%d\t ", arr[i]);
}
```

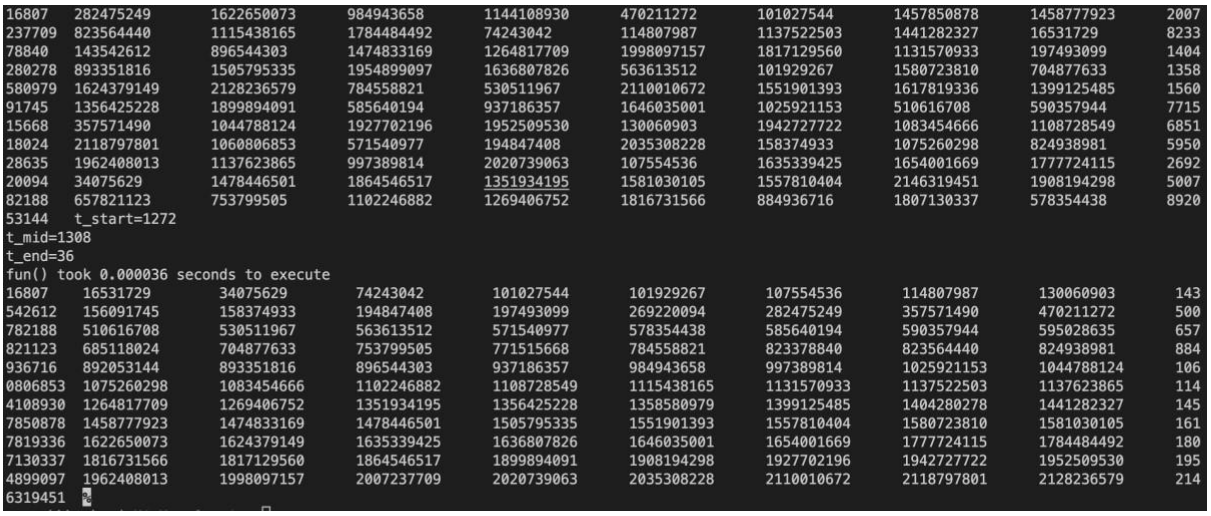
OUTPUT:

Sorted data



```
0       1       2       3       4       5       6       7       8       9       10      11      12      13      14      15      16      17 1
8       19      20      21      22      23      24      25      26      27      28      29      30      31      32      33      34      35 3
6       37      38      39      40      41      42      43      44      45      46      47      48      49      50      51      52      53 5
4       55      56      57      58      59      60      61      62      63      64      65      66      67      68      69      70      71 7
2       73      74      75      76      77      78      79      80      81      82      83      84      85      86      87      88      89 9
0       91      92      93      94      95      96      97      98      99      t_start=1478
t_mid=1500
t_end=22
fun() took 0.000022 seconds to execute
0        1        2        3        4        5        6        7        8        9        10       11       12       13       14       15       16       17
18       19       20       21       22       23       24       25       26       27       28       29       30       31       32       33       34       35
36       37       38       39       40       41       42       43       44       45       46       47       48       49       50       51       52       53
54       55       56       57       58       59       60       61       62       63       64       65       66       67       68       69       70       71
72       73       74       75       76       77       78       79       80       81       82       83       84       85       86       87       88       89
90       91       92       93       94       95       96       97       98       99       %
```

Random data



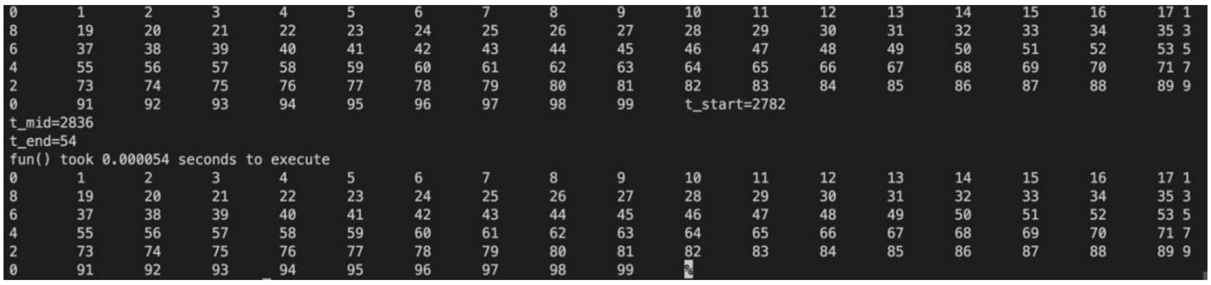
```
16807   282475249       1622650073      984943658       1144108930      470211272       101027544       1457850878      1458777923      2007
237709  823564440       1115438165      1784484492      74243042        114807987       1137522503      1441282327      16531729        8233
78840   143542612       896544303       1474833169      1264817709      1998097157      1817129560      1131570933      197493099       1404
280278  893351816       1505795335      1954899097      1636807826      563613512       101929267       1580723810      704877633       1358
580979  1624379149      2128236579      784558821       530511967       2110010672      1551901393      1617819336      1399125485      1560
91745   1356425228      1899894091      585640194       937186357       1646035001      1025921153      510616708       590357944       7715
15668   357571490       1044788124      1927702196      1952509530      130060903       1942727722      1083454666      1108728549      6851
18024   2118797801      1060806853      571540977       194847408       2035308228      158374933       1075260298      824938981       5950
28635   1962408013      1137623865      997389814       2020739063      107554536       1635339425      1654001669      1777724115      2692
20094   34075629        1478446501      1864546517      1351934195      1581030105      1557810404      2146319451      1908194298      5007
82188   657821123       753799505       1102246882      1269406752      1816731566      884936716       1807130337      578354438       8920
53144   t_start=1272
t_mid=1308
t_end=36
fun() took 0.000036 seconds to execute
16807    16531729        34075629        74243042        101027544       101929267       107554536       114807987       130060903       143
542612   156091745       158374933       194847408       197493099       269220094       282475249       357571490       470211272       500
782188   510616708       530511967       563613512       571540977       578354438       585640194       590357944       595028635       657
821123   685118024       704877633       753799505       771515668       784558821       823378840       823564440       824938981       884
936716   892053144       893351816       896544303       937186357       984943658       997389814       1025921153      1044788124      106
0806853  1075260298      1083454666      1102246882      1108728549      1115438165      1131570933      1137522503      1137623865      114
4108930  1264817709      1269406752      1351934195      1356425228      1358580979      1399125485      1404280278      1441282327      145
7850878  1458777923      1474833169      1478446501      1505795335      1551901393      1557810404      1580723810      1581030105      161
7819336  1622650073      1624379149      1635339425      1636807826      1646035001      1654001669      1777724115      1784484492      180
7130337  1816731566      1817129560      1864546517      1899894091      1908194298      1927702196      1942727722      1952509530      195
4899097  1962408013      1998097157      2007237709      2020739063      2035308228      2110010672      2118797801      2128236579      214
6319451  %
```

2. #include<stdio.h>
```
#include<time.h>


void bubble(int arr[], int n);


int main(){
int n=100;     int
arr[n];

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

    clock_t t_start,t_end,t_mid;       t_start = clock();
//starting time after initializing data
printf("t_start=%lu\n",t_start);

    bubble(arr, n);
          t_mid=clock();                                              //mid time after function
call     printf("t_mid=%lu\n",t_mid);

    t_end = t_mid - t_start;                                     //ending time after mid - start
printf("t_end=%lu\n",t_end);
```

```
    double time_taken = ((double)t_end)/CLOCKS_PER_SEC;           //clock per sec is a macro
and 1cps= 1 million micro sec
    printf("fun() took %f seconds to execute \n", time_taken);

    return 0;

}

void bubble(int arr[], int n) // function to implement bubble sort
 {
   int i, j, temp;
   for(i = 0; i < n; i++)
    {
      for(j = i+1; j < n; j++)
        {
            if(arr[j] < arr[i])
            {
temp = arr[i];
arr[i] = arr[j];
arr[j] = temp;
            }
        }
    }
 }
```

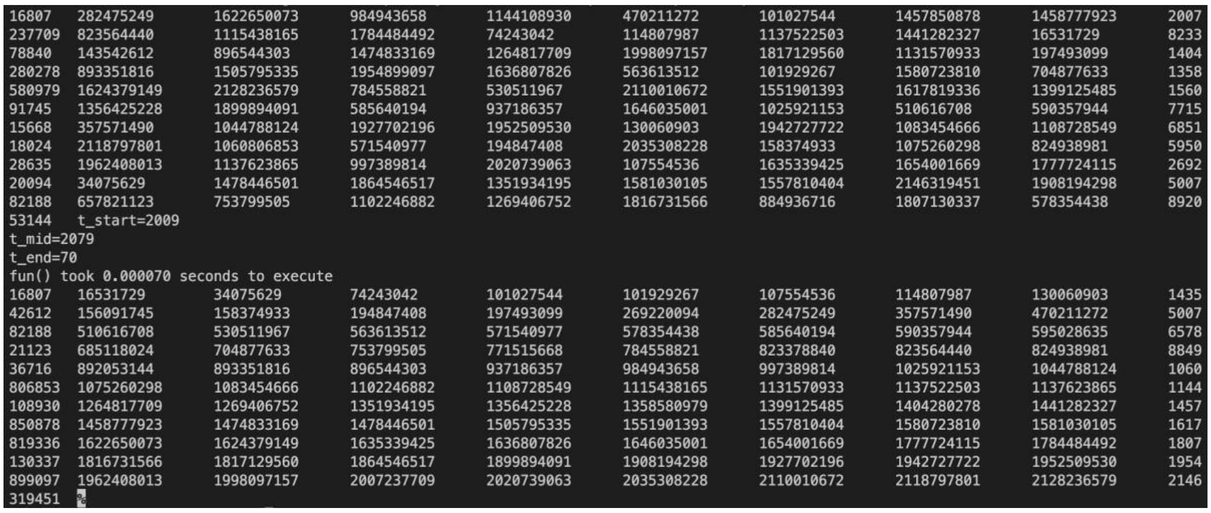
OUTPUT:

Sorted data



```
0       1       2       3       4       5       6       7       8       9       10      11      12      13      14      15      16      17 1
8       19      20      21      22      23      24      25      26      27      28      29      30      31      32      33      34      35 3
6       37      38      39      40      41      42      43      44      45      46      47      48      49      50      51      52      53 5
4       55      56      57      58      59      60      61      62      63      64      65      66      67      68      69      70      71 7
2       73      74      75      76      77      78      79      80      81      82      83      84      85      86      87      88      89 9
0       91      92      93      94      95      96      97      98      99      t_start=2782
t_mid=2836
t_end=54
fun() took 0.000054 seconds to execute
0       1       2       3       4       5       6       7       8       9       10      11      12      13      14      15      16      17 1
8       19      20      21      22      23      24      25      26      27      28      29      30      31      32      33      34      35 3
6       37      38      39      40      41      42      43      44      45      46      47      48      49      50      51      52      53 5
4       55      56      57      58      59      60      61      62      63      64      65      66      67      68      69      70      71 7
2       73      74      75      76      77      78      79      80      81      82      83      84      85      86      87      88      89 9
0       91      92      93      94      95      96      97      98      99      %
```

Random data

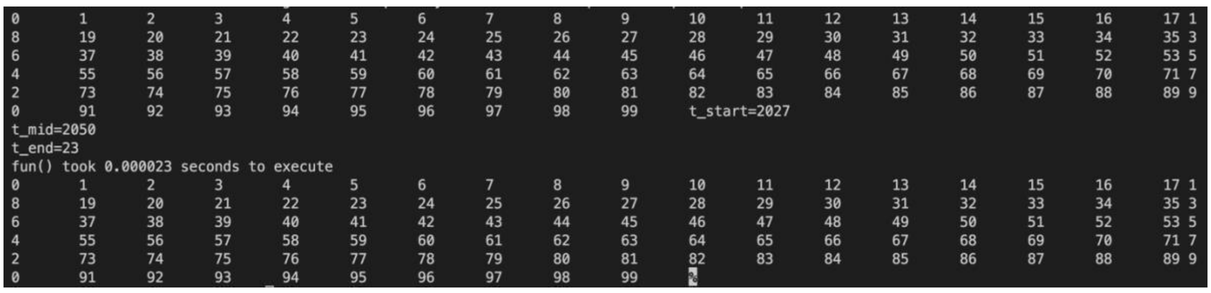


```
16807   282475249       1622650073      984943658       1144108930      470211272       101027544       1457850878      1458777923      2007
237709  823564440       1115438165      1784484492      74243042        114807987       1137522503      1441282327      16531729        8233
78840   143542612       896544303       1474833169      1264817709      1998097157      1817129560      1131570933      197493099       1404
280278  893351816       1505795335      1954899097      1636807826      563613512       101929267       1580723810      704877633       1358
580979  1624379149      2128236579      784558821       530511967       2110010672      1551901393      1617819336      1399125485      1560
91745   1356425228      1899894091      585640194       937186357       1646035001      1025921153      510616708       590357944       7715
15668   357571490       1044788124      1927702196      1952509530      130060903       1942727722      1083454666      1108728549      6851
18024   2118797801      1060806853      571540977       194847408       2035308228      158374933       1075260298      824938981       5950
28635   1962408013      1137623865      997389814       2020739063      107554536       1635339425      1654001669      1777724115      2692
20094   34075629        1478446501      1864546517      1351934195      1581030105      1557810404      2146319451      1908194298      5007
82188   657821123       753799505       1102246882      1269406752      1816731566      884936716       1807130337      578354438       8920
53144   t_start=2009
t_mid=2079
t_end=70
fun() took 0.000070 seconds to execute
16807   16531729        34075629        74243042        101027544       101929267       107554536       114807987       130060903       1435
42612   156091745       158374933       194847408       197493099       269220094       282475249       357571490       470211272       5007
82188   510616708       530511967       563613512       571540977       578354438       585640194       590357944       595028635       6578
21123   685118024       704877633       753799505       771515668       784558821       823378840       823564440       824938981       8849
36716   892053144       893351816       896544303       937186357       984943658       997389814       1025921153      1044788124      1060
806853  1075260298      1083454666      1102246882      1108728549      1115438165      1131570933      1137522503      1137623865      1144
108930  1264817709      1269406752      1351934195      1356425228      1358580979      1399125485      1404280278      1441282327      1457
850878  1458777923      1474833169      1478446501      1505795335      1551901393      1557810404      1580723810      1581030105      1617
819336  1622650073      1624379149      1635339425      1636807826      1646035001      1654001669      1777724115      1784484492      1807
130337  1816731566      1817129560      1864546517      1899894091      1908194298      1927702196      1942727722      1952509530      1954
899097  1962408013      1998097157      2007237709      2020739063      2035308228      2110010672      2118797801      2128236579      2146
319451  %
```

3.
```
#include<stdio.h>
#include<time.h>
#include <stdlib.h>

void insertion(int arr[], int n);
void printArr(int arr[], int n);

int main(){     int
n=100;
    int arr[n];

    for(int i=0;i<n;i++){
arr[i]=rand();
    }

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

    clock_t t_start,t_end,t_mid;      t_start = clock();
//starting time after initializing data
printf("t_start=%lu\n",t_start);

    insertion(arr, n);
        t_mid=clock();                                    //mid time after function
call     printf("t_mid=%lu\n",t_mid);

    t_end = t_mid - t_start;                              //ending time after mid - start
printf("t_end=%lu\n",t_end);

    double time_taken = ((double)t_end)/CLOCKS_PER_SEC;          //clock per sec is a macro
and 1cps= 1 million micro sec
    printf("fun() took %f seconds to execute \n", time_taken);
    printArr(arr, n);

    return 0;

}


void insertion(int arr[], int n) /* function to sort an aay with insertion sort */
{
    int i, j, temp;       for (i =
1; i < n; i++) {           temp
= arr[i];          j = i
- 1;

        while(j>=0 && temp <= arr[j])  /* Move the elements greater than temp to one position
ahead from their current position*/
        {
            arr[j+1] = arr[j];
            j = j-1;
        }
        arr[j+1] = temp;
    }
}

void printArr(int arr[], int n) /* function to print the array */
{       int
i;
    for (i = 0; i < n; i++)
printf("%d\t", arr[i]);
}
```

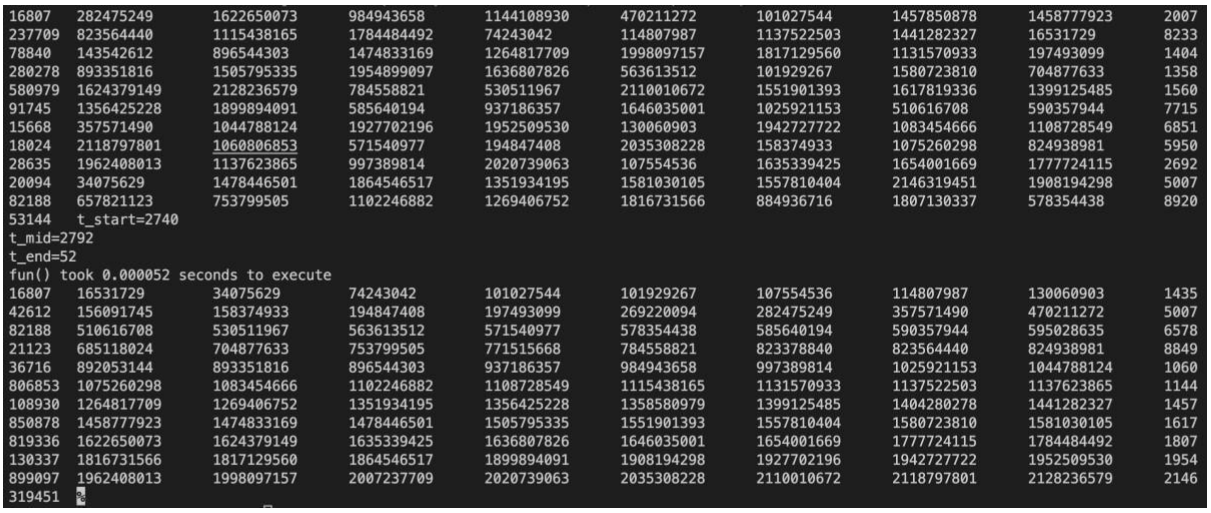
OUTPUT:

Sorted data



```
0       1       2       3       4       5       6       7       8       9       10      11      12      13      14      15      16      17 1
8       19      20      21      22      23      24      25      26      27      28      29      30      31      32      33      34      35 3
6       37      38      39      40      41      42      43      44      45      46      47      48      49      50      51      52      53 5
4       55      56      57      58      59      60      61      62      63      64      65      66      67      68      69      70      71 7
2       73      74      75      76      77      78      79      80      81      82      83      84      85      86      87      88      89 9
0       91      92      93      94      95      96      97      98      99      t_start=2027
t_mid=2050
t_end=23
fun() took 0.000023 seconds to execute
0       1       2       3       4       5       6       7       8       9       10      11      12      13      14      15      16      17 1
8       19      20      21      22      23      24      25      26      27      28      29      30      31      32      33      34      35 3
6       37      38      39      40      41      42      43      44      45      46      47      48      49      50      51      52      53 5
4       55      56      57      58      59      60      61      62      63      64      65      66      67      68      69      70      71 7
2       73      74      75      76      77      78      79      80      81      82      83      84      85      86      87      88      89 9
0       91      92      93      94      95      96      97      98      99      %
```

Random data



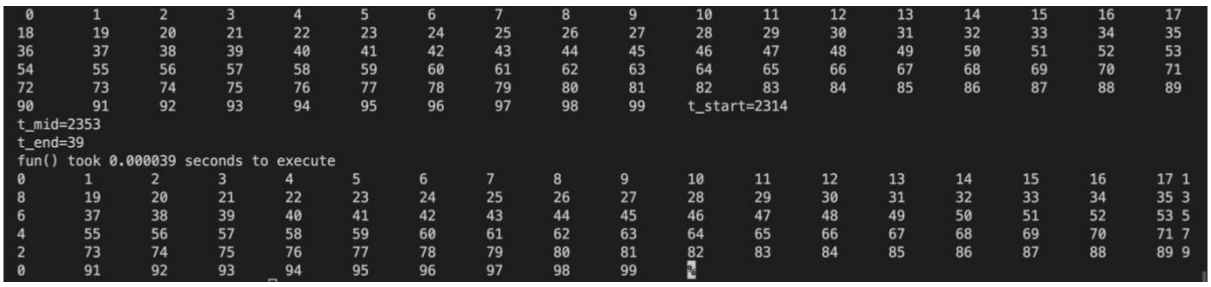
```
16807   282475249       1622650073      984943658       1144108930      470211272       101027544       1457850878      1458777923      2007
237709  823564440       1115438165      1784484492      74243042        114807987       1137522503      1441282327      16531729        8233
78840   143542612       896544303       1474833169      1264817709      1998097157      1817129560      1131570933      197493099       1404
280278  893351816       1505795335      1954899097      1636807826      563613512       101929267       1580723810      704877633       1358
580979  1624379149      2128236579      784558821       530511967       2110010672      1551901393      1617819336      1399125485      1560
91745   1356425228      1899894091      585640194       937186357       1646035001      1025921153      510616708       590357944       7715
15668   357571490       1044788124      1927702196      1952509530      130060903       1942727722      1083454666      1108728549      6851
18024   2118797801      1060806853      571540977       194847408       2035308228      158374933       1075260298      824938981       5950
28635   1962408013      1137623865      997389814       2020739063      107554536       1635339425      1654001669      1777724115      2692
20094   34075629        1478446501      1864546517      1351934195      1581030105      1557810404      2146319451      1908194298      5007
82188   657821123       753799505       1102246882      1269406752      1816731566      884936716       1807130337      578354438       8920
53144   t_start=2740
t_mid=2792
t_end=52
fun() took 0.000052 seconds to execute
16807   16531729        34075629        74243042        101027544       101929267       107554536       114807987       130060903       1435
42612   156091745       158374933       194847408       197493099       269220094       282475249       357571490       470211272       5007
82188   510616708       530511967       563613512       571540977       578354438       585640194       590357944       595028635       6578
21123   685118024       704877633       753799505       771515668       784558821       823378840       823564440       824938981       8849
36716   892053144       893351816       896544303       937186357       984943658       997389814       1025921153      1044788124      1060
806853  1075260298      1083454666      1102246882      1108728549      1115438165      1131570933      1137522503      1137623865      1144
108930  1264817709      1269406752      1351934195      1356425228      1358580979      1399125485      1404280278      1441282327      1457
850878  1458777923      1474833169      1478446501      1505795335      1551901393      1557810404      1580723810      1581030105      1617
819336  1622650073      1624379149      1635339425      1636807826      1646035001      1654001669      1777724115      1784484492      1807
130337  1816731566      1817129560      1864546517      1899894091      1908194298      1927702196      1942727722      1952509530      1954
899097  1962408013      1998097157      2007237709      2020739063      2035308228      2110010672      2118797801      2128236579      2146
319451  %
```

4.

```
#include<stdio.h>
#include<time.h>
#include <stdlib.h>



void merge(int arr[], int beg, int mid, int end) ;
void mergeSort(int arr[], int beg, int end); void
printArr(int arr[], int n);

int main(){     int
n=100;
    int arr[n],i;

    for(i=0;i<n;i++){
arr[i]=rand();
    }

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

    clock_t t_start,t_end,t_mid;      t_start = clock();
//starting time after initializing data
printf("t_start=%lu\n",t_start);
```

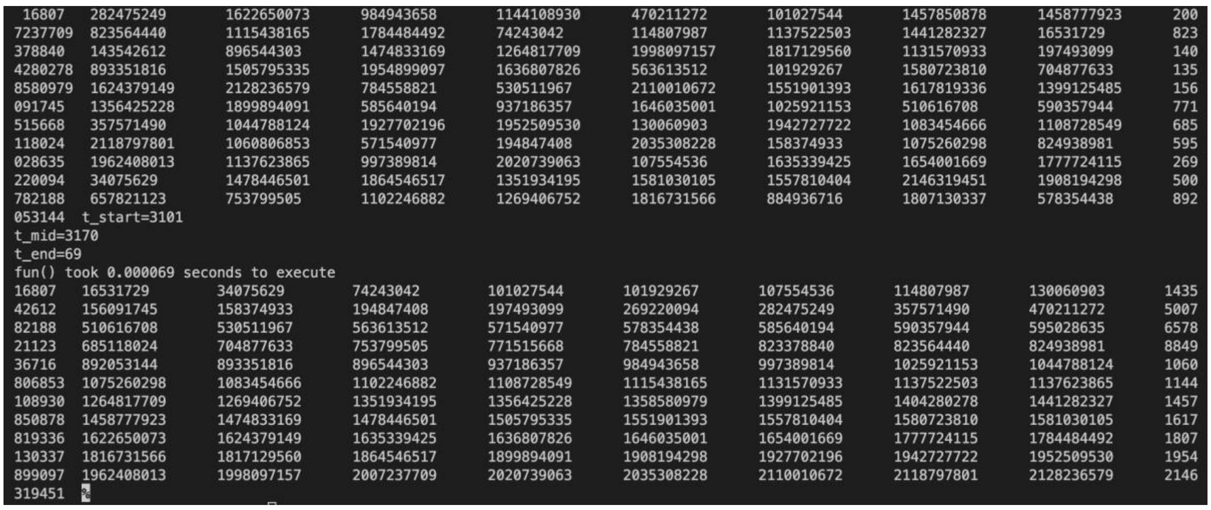
```
    mergeSort(arr, 0, n-1);
          t_mid=clock();                                   //mid time after function
call     printf("t_mid=%lu\n",t_mid);

    t_end = t_mid - t_start;                               //ending time after mid - start
printf("t_end=%lu\n",t_end);

    double time_taken = ((double)t_end)/CLOCKS_PER_SEC;          //clock per sec is a macro
and 1cps= 1 million micro sec
    printf("fun() took %f seconds to execute \n", time_taken);

    printArr(arr, n);

    return 0;

}

void mergeSort(int a[], int beg, int end)
{
    if (beg < end)
    {
        int mid = (beg + end) / 2;
        mergeSort(a, beg, mid);
mergeSort(a, mid + 1, end);
        merge(a, beg, mid, end);
    }
}


void merge(int arr[], int beg, int mid, int end)
{
    int i, j, k;       int n1 =
mid - beg + 1;         int n2
= end - mid;

    int LeftArray[n1], RightArray[n2]; //temporary arrays

    /* copy data to temp arrays */
for (int i = 0; i < n1; i++)
LeftArray[i] = arr[beg + i];         for
(int j = 0; j < n2; j++)
RightArray[j] = arr[mid + 1 + j];

    i = 0, /* initial index of first sub-array */       j
= 0; /* initial index of second sub-array */
    k = beg;  /* initial index of merged sub-array */
    while (i < n1 && j < n2)
    {
        if(LeftArray[i] <= RightArray[j])
        {
            arr[k] = LeftArray[i];
            i++;
}              else
        {
            arr[k] =
RightArray[j];                  j++;
}              k++;
    }         while
(i<n1)
    {
        arr[k] = LeftArray[i];
i++;
k++;
    }

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

void printArr(int arr[], int n) /* function to print the array */
{       int
i;
    for (i = 0; i < n; i++)
printf("%d\t", arr[i]);
}
```

Sorted data



```
 0       1       2       3       4       5       6       7       8       9       10      11      12      13      14      15      16      17
18      19      20      21      22      23      24      25      26      27      28      29      30      31      32      33      34      35
36      37      38      39      40      41      42      43      44      45      46      47      48      49      50      51      52      53
54      55      56      57      58      59      60      61      62      63      64      65      66      67      68      69      70      71
72      73      74      75      76      77      78      79      80      81      82      83      84      85      86      87      88      89
90      91      92      93      94      95      96      97      98      99      t_start=2314
t_mid=2353
t_end=39
fun() took 0.000039 seconds to execute
0       1       2       3       4       5       6       7       8       9       10      11      12      13      14      15      16      17 1
8       19      20      21      22      23      24      25      26      27      28      29      30      31      32      33      34      35 3
6       37      38      39      40      41      42      43      44      45      46      47      48      49      50      51      52      53 5
4       55      56      57      58      59      60      61      62      63      64      65      66      67      68      69      70      71 7
2       73      74      75      76      77      78      79      80      81      82      83      84      85      86      87      88      89 9
0       91      92      93      94      95      96      97      98      99      %
```

Random data



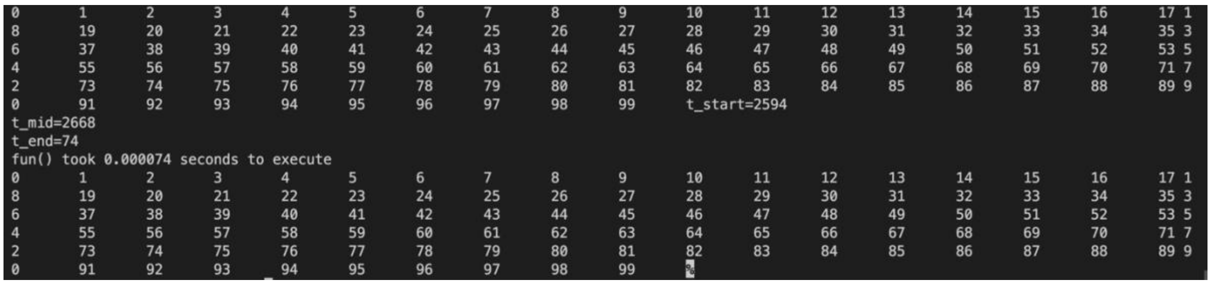
```
 16807   282475249      1622650073      984943658       1144108930      470211272       101027544       1457850878      1458777923      200
7237709  823564440      1115438165      1784484492      74243042        114807987       1137522503      1441282327      16531729        823
378840   143542612      896544303       1474833169      1264817709      1998097157      1817129560      1131570933      197493099       140
4280278  893351816      1505795335      1954899097      1636807826      563613512       101929267       1580723810      704877633       135
8580979  1624379149     2128236579      784558821       530511967       2110010672      1551901393      1617819336      1399125485      156
091745   1356425228     1899894091      585640194       937186357       1646035001      1025921153      510616708       590357944       771
515668   357571490      1044788124      1927702196      1952509530      130060903       1942727722      1083454666      1108728549      685
118024   2118797801     1060806853      571540977       194847408       2035308228      158374933       1075260298      824938981       595
028635   1962408013     1137623865      997389814       2020739063      107554536       1635339425      1654001669      1777724115      269
220094   34075629       1478446501      1864546517      1351934195      1581030105      1557810404      2146319451      1908194298      500
782188   657821123      753799505       1102246882      1269406752      1816731566      884936716       1807130337      578354438       892
053144  t_start=3101
t_mid=3170
t_end=69
fun() took 0.000069 seconds to execute
16807   16531729        34075629        74243042        101027544       101929267       107554536       114807987       130060903       1435
42612   156091745       158374933       194847408       197493099       269220094       282475249       357571490       470211272       5007
82188   510616708       530511967       563613512       571540977       578354438       585640194       590357944       595028635       6578
21123   685118024       704877633       753799505       771515668       784558821       823378840       823564440       824938981       8849
36716   892053144       893351816       896544303       937186357       984943658       997389814       1025921153      1044788124      1060
806853  1075260298      1083454666      1102246882      1108728549      1115438165      1131570933      1137522503      1137623865      1144
108930  1264817709      1269406752      1351934195      1356425228      1358580979      1399125485      1404280278      1441282327      1457
850878  1458777923      1474833169      1478446501      1505795335      1551901393      1557810404      1580723810      1581030105      1617
819336  1622650073      1624379149      1635339425      1636807826      1646035001      1654001669      1777724115      1784484492      1807
130337  1816731566      1817129560      1864546517      1899894091      1908194298      1927702196      1942727722      1952509530      1954
899097  1962408013      1998097157      2007237709      2020739063      2035308228      2110010672      2118797801      2128236579      2146
319451  %
```

5.

```
#include<stdio.h>
#include<time.h>
#include <stdlib.h>

void quick(int a[], int start, int end);  void
printArr(int arr[], int n);

int main(){
int n=100;     int
arr[n];

    for(int i=0;i<n;i++){
        arr[i]=i;
    }

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

    clock_t t_start,t_end,t_mid;      t_start = clock();
//starting time after initializing data
printf("t_start=%lu\n",t_start);

    quick(arr, 0, n - 1);
          t_mid=clock();                                   //mid time after function
call     printf("t_mid=%lu\n",t_mid);

    t_end = t_mid - t_start;                               //ending time after mid - start
printf("t_end=%lu\n",t_end);

    double time_taken = ((double)t_end)/CLOCKS_PER_SEC;           //clock per sec is a macro
and 1cps= 1 million micro sec
```

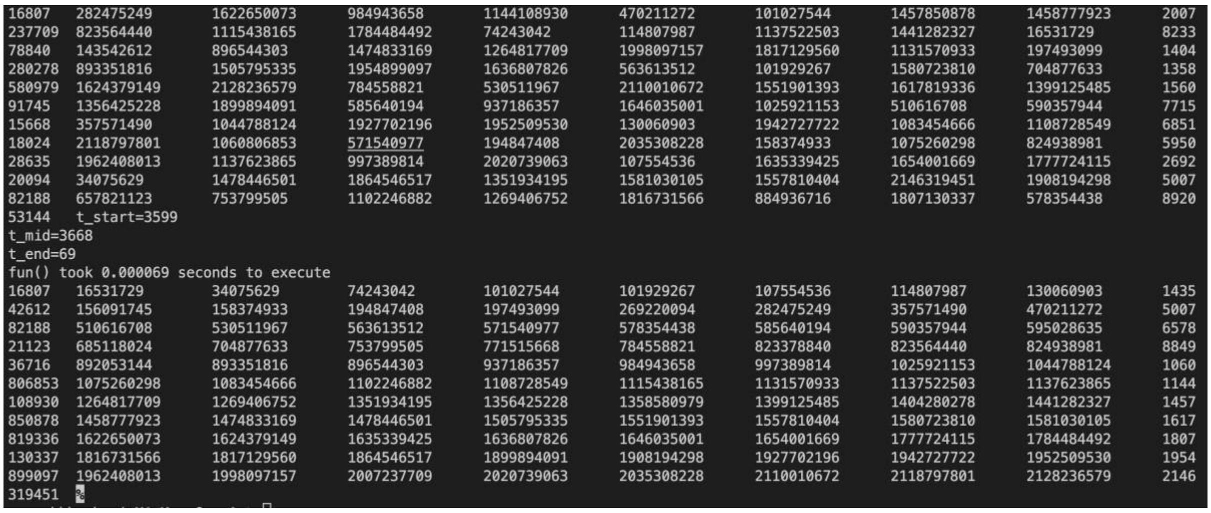
```
    printf("fun() took %f seconds to execute \n", time_taken);

    printArr(arr, n);
return 0;

}


int partition (int a[], int start, int end)
{
    int pivot = a[end]; // pivot element
    int i = (start - 1);

    for (int j = start; j <= end - 1; j++)
    {
        // If current element is smaller than the pivot           if
(a[j] < pivot)
        {
            i++; // increment index of smaller element
int t = a[i];               a[i] = a[j];               a[j] = t;
}       }        int t = a[i+1];       a[i+1] = a[end];
a[end] = t;       return
(i + 1);
}

/* function to implement quick sort */    void quick(int a[], int start, int end) /* a[] = array
to be sorted, start = Starting index, end = Ending index */
{
    if (start < end)
    {           int p = partition(a, start, end); //p is the
partitioning index
quick(a, start, p - 1);
        quick(a, p + 1, end);
    }
}


void printArr(int arr[], int n) /* function to print the array */
{       int
i;
    for (i = 0; i < n; i++)
printf("%d\t", arr[i]);
}
```

Sorted data



```
0       1       2       3       4       5       6       7       8       9       10      11      12      13      14      15      16      17 1
8       19      20      21      22      23      24      25      26      27      28      29      30      31      32      33      34      35 3
6       37      38      39      40      41      42      43      44      45      46      47      48      49      50      51      52      53 5
4       55      56      57      58      59      60      61      62      63      64      65      66      67      68      69      70      71 7
2       73      74      75      76      77      78      79      80      81      82      83      84      85      86      87      88      89 9
0       91      92      93      94      95      96      97      98      99      t_start=2594
t_mid=2668
t_end=74
fun() took 0.000074 seconds to execute
0       1       2       3       4       5       6       7       8       9       10      11      12      13      14      15      16      17 1
8       19      20      21      22      23      24      25      26      27      28      29      30      31      32      33      34      35 3
6       37      38      39      40      41      42      43      44      45      46      47      48      49      50      51      52      53 5
4       55      56      57      58      59      60      61      62      63      64      65      66      67      68      69      70      71 7
2       73      74      75      76      77      78      79      80      81      82      83      84      85      86      87      88      89 9
0       91      92      93      94      95      96      97      98      99      %
```

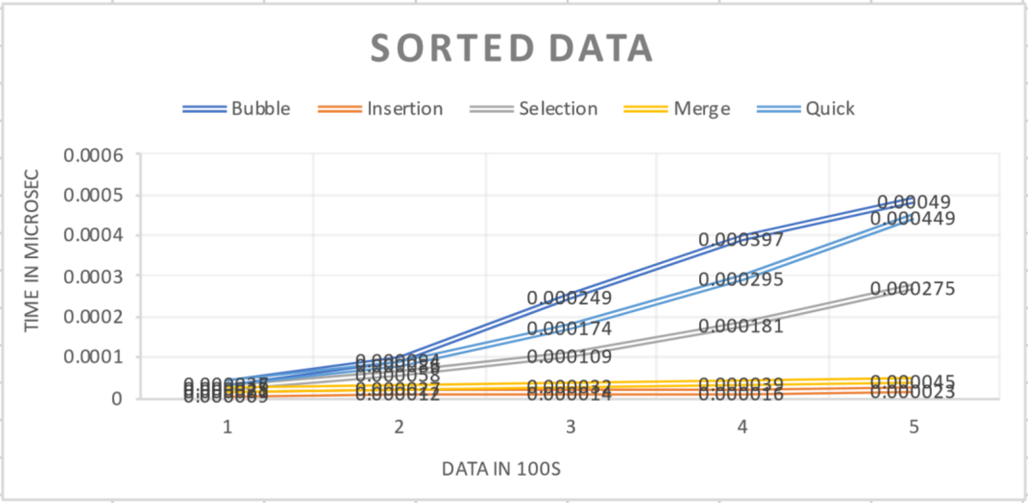
Random data



```
16807   282475249       1622650073      984943658       1144108930      470211272       101027544       1457850878      1458777923      2007
237709  823564440       1115438165      1784484492      74243042        114807987       1137522503      1441282327      16531729        8233
78840   143542612       896544303       1474833169      1264817709      1998097157      1817129560      1131570933      197493099       1404
280278  893351816       1505795335      1954899097      1636807826      563613512       101929267       1580723810      704877633       1358
580979  1624379149      2128236579      784558821       530511967       2110010672      1551901393      1617819336      1399125485      1560
91745   1356425228      1899894091      585640194       937186357       1646035001      1025921153      510616708       590357944       7715
15668   357571490       1044788124      1927702196      1952509530      130060903       1942727722      1083454666      1108728549      6851
18024   2118797801      1060806853      571540977       194847408       2035308228      158374933       1075260298      824938981       5950
28635   1962408013      1137623865      997389814       2020739063      107554536       1635339425      1654001669      1777724115      2692
20094   34075629        1478446501      1864546517      1351934195      1581030105      1557810404      2146319451      1908194298      5007
82188   657821123       753799505       1102246882      1269406752      1816731566      884936716       1807130337      578354438       8920
53144   t_start=3599
t_mid=3668
t_end=69
fun() took 0.000069 seconds to execute
16807   16531729        34075629        74243042        101027544       101929267       107554536       114807987       130060903       1435
42612   156091745       158374933       194847408       197493099       269220094       282475249       357571490       470211272       5007
82188   510616708       530511967       563613512       571540977       578354438       585640194       590357944       595028635       6578
21123   685118024       704877633       753799505       771515668       784558821       823378840       823564440       824938981       8849
36716   892053144       893351816       896544303       937186357       984943658       997389814       1025921153      1044788124      1060
806853  1075260298      1083454666      1102246882      1108728549      1115438165      1131570933      1137522503      1137623865      1144
108930  1264817709      1269406752      1351934195      1356425228      1358580979      1399125485      1404280278      1441282327      1457
850878  1458777923      1474833169      1478446501      1505795335      1551901393      1557810404      1580723810      1581030105      1617
819336  1622650073      1624379149      1635339425      1636807826      1646035001      1654001669      1777724115      1784484492      1807
130337  1816731566      1817129560      1864546517      1899894091      1908194298      1927702196      1942727722      1952509530      1954
899097  1962408013      1998097157      2007237709      2020739063      2035308228      2110010672      2118797801      2128236579      2146
319451  %
```

GRAPH:

total data in asc		bubble	insertion	selection	merge	quick
100		0.000037	0.000009	0.000025	0.000021	0.000036
200		0.000094	0.000012	0.000058	0.000027	0.000086
300		0.000249	0.000014	0.000109	0.000032	0.000174
400		0.000397	0.000016	0.000181	0.000039	0.000295
500		0.00049	0.000023	0.000275	0.000045	0.000449


SORTED DATA
Bubble
Insertion
Selection
Merge
Quick
TIME IN MICROSEC
DATA IN 100S