1) Sort a given set of elements using the Quicksort 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 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 Exch(int *p, int *q) {
   int temp = *p;
   *p = *q;
   *q = temp;
}

int partition(int a[], int low, int high) {
   int key = low;
   int i = low + 1;
   int j = high;

   while (i <= j) {
      while (a[i] <= a[key])
         i = i + 1;
      while (a[j] > a[key])
         j = j - 1;
      if (i < j)
         Exch(&a[i], &a[j]);
   }
   Exch(&a[j], &a[key]);
   return j;
}

void QuickSort(int a[], int low, int high) {
   int i, j, key, k;
   if (low >= high)
      return;
   j = partition(a, low, high);
   QuickSort(a, low, j - 1);
   QuickSort(a, j + 1, high);
}
```

```
int main() {
 int n, a[100], k;
 clock_t st, et;
 double ts;

 printf("\n Enter How many Numbers: ");
 scanf("%d", &n);
 printf("\nThe Random Numbers are:\n");

 for (k = 1; k <= n; k++) {
// generates random number between 50 and 100
 a[k] = rand() % (100 - 50 + 1) + 50;
 printf("%d\t", a[k]);
 }

 st = clock();
 for (k = 1; k < 10000; k++)
 QuickSort(a, 1, n);
 et = clock(); // approximate processor time that is consumed
 ts = (double)(et - st) / CLOCKS_PER_SEC;
 ts = ts / 10000;

 printf("\nSorted Numbers are: \n ");
 for (k = 1; k <= n; k++)
 printf("%d\t", a[k]);

 printf("\nThe time taken is %e sec", ts);

 return 0;
}
```

```
 Enter How many Numbers: 7

The Random Numbers are:
78      69      55      88      72      94      62
Sorted Numbers are:
 55     62      69      72      78      88      94
The time taken is 2.385000e-07 sec%
```