

BITS Pilani, Pilani Campus  
2<sup>nd</sup> Sem. 2020-21  
CS F211 Data Structures & Algorithms

=====

Lab VII

=====

**Topics:** QuickSort, Performance Measurements - Running Time and Space Usage, Command Line Arguments

**Programming Environment:** C on Linux

In all the exercises in this lab sheet, you must construct an array of employee records containing the following fields – {*name* and *empID*}, where *name* is a string containing a maximum of 10 characters and *empID* is an integer value containing the employee ID. You must use *empID* as the key for all comparisons in your sorting algorithms. Sample input files are given along with this sheet, which can be used in the exercises as input files. Note that all input files have the fields *name* and *empID* separated by space.

**Exercise 1: [Expected Time: 100 minutes.]**

- a) Implement the iterative version of QuickSort with an explicit stack such that sub-lists of size less than or equal to *S* are left untouched (i.e. not sorted). *S* is passed as a parameter to QuickSort. Note that  $S < 1$  should result in QuickSort running completely i.e. all sub-lists get sorted.
- b) Implement the iterative version of Insertion Sort algorithm.
- c) Implement a sorting procedure that (i) invokes QuickSort (your solution to (a)) on the input list, with a cutoff value for size, followed by (ii) an insertion sort of the entire list.

**Exercise 2: [Expected Time: 30 minutes]**

- a) Write a *bisection* procedure to decide the cutoff size, above which QuickSort is faster than insertion sort:

```
define testRun(Ls, size) {  
    run InsertionSort and QuickSort – separately – on Ls;  
    return the running time measurements (IStime, QStime);  
}  
  
define estimateCutoff {  
    tt1 = testRun(Ls, min); // assert tt1.IStime < tt1.QStime  
    tt2 = testRun(Ls, max); // assert tt1.IStime > tt1.QStime  
  
    repeat { mid = (min+max)/2;  
        tt = testRun(Ls, mid);  
        use tt to narrow the range (i.e. min=mid or max=mid)  
    } (until tt.IStime approximately== tt.QStime)  
    return mid;  
}
```

- b) Write a main function in C that takes an input file name, an output file name as command line arguments and then:
- i. Runs estimateCutoff to decide the cutoff size;
  - ii. For  $N=10^4, 10^5, 10^6, \dots 10^9$ :
    - Read N values from input file
    - Call sorting procedure in 1.c) with cutoff size as estimated in (i).
    - Measure the time taken for sorting and store it
    - Write the sorted list into the output file.

**Exercise 3: [45 minutes]**

Repeat 2. (b) for a few different input files and plot curves (one per input file) of size versus time taken. Do curve fitting to find out the closest defining function per curve and match it with estimated time complexity.

**Assignment with respect to this lab:** Solve Exercise 3 and upload the solution. There might be issues with your system while varying N (in 2(b) above) from  $10^4$  to  $10^9$ . In such case, you can put upper limit on the value of N. Also, you need not to do curve fitting as a part of this assignment.

In addition to all the .c files, your submission should have one file, named complete.c, in which all the functions you have written must be there. Just copy all the functions and main in this "complete.c" file.