National Institute of Technology, Calicut
Department of Computer Science and Engineering
CS2094 – Data Structures Lab
Assignment 2

Date of Submission: On or before **22-02-2014(Sunday) 5:00pm**

About the Assignment

This assignment is based on Sorting and Searching. Implement ALL questions. Your program MUST take input from text files, which is formatted as specified below, and/ or with each question.

Measure the running time of your algorithms (*use the time measuring options available, in the programming language you are using to implement the algorithms*). Compare the variation of running time of each algorithm, corresponding to the variation in the size of the input.

Each question should be tested against input files that contain various number of data items. For Insertion sort, Merge sort, Quick Sort and Heap Sort the number of data items should be of the order of 10^i, where i = 1, 2, 3, …, 8.

Common Input Format

Searching

The first line of the input file contains a positive integer ***n***, the number of data items.
Then, ***n*** lines follow, each line containing exactly one data item. (In sorted order for Binary search)
Next line contains the number to be searched.

Sorting

The first line of the input file contains a positive integer ***n***, the number of data items to be sorted.
Then, ***n*** lines follow, each line containing exactly one data item.

Common Output Format

Sorting

The (sorted) output must contain exactly ***n+1*** lines.
The first ***n*** lines must contain exactly one data item.
The last line must contain a single line, that prints the measured running time of the sorting algorithm, in the format as in the example given here: "**Running time: 2.013 sec**".

Standard of Conduct

Violations of academic integrity will be severely penalized. Each student is expected to adhere to high standards of ethical conduct, especially those related to cheating and plagiarism. Any submitted work MUST BE an individual effort. Any academic dishonesty will result in zero marks in the corresponding exam or evaluation and will be reported to the department council for record keeping and for permission to assign F grade in the course. The department policy on academic integrity can be found at: http://cse.nitc.ac.in/sites/default/files/Academic-Integrity.pdf.

General Instructions for all the questions:

1) Invalid input should be detected and suitable error messages should be generated.

2) Sample inputs are just indicative.

1 **Linear Search**

Given an array of ***n*** integers, write a program to search whether a particular integer, say ***k***, is present in the array. If present, return the index of the same. Otherwise, return -1. Implement linear search using recursion.

Example

Input:
7
18
45
56
12
34
87
14
56

Output:
2

2. Binary Search

Given a sorted array of n integers that has been rotated an unknown number of times, give an O(log n) algorithm that finds an element in the array. You may assume that the array was originally sorted in increasing order.

Example

Input:
10
15
16
19
20
25
1
3
4
5
7
5

Output:
8

3. Insertion Sort

Let A[1. . . n] be an array of n integers. If i < j and A[i] > A[j], then the pair (i,j) is called an inversion of A. Implement the *insertion sort* algorithm to sort the given array A, as well as to count the number of inversions.

Input: An array of n integers, <a_1,a_2,...,a_n>

Output: The *n* integers in sorted order <a_1',a_2',……a_n'> such that a_1'≤a_2'≤…≤a_n' and the number of inversions.

Sample Input

6
712
-45
456765
0
-8907
4566

4. Selection Sort

Given a list of ***n* words**, implement the selection sort algorithm, to sort them *lexicographical order.* The words contain only lower case English letters (**a – z**).

Input: An array of ***n*** words consisting of only lower case English letters .
Output: The n words sorted in lexicographical order.

Sample Input

6
nitc
sorting
datastructures
sort
select
insert

5. Quick Sort

Implement the *quick sort* algorithm, to sort ***n* real numbers**.

Input: An array on ***n*** real numbers $<a_1,a_2,\ldots\ldots a_n>$.
Output: The n real numbers in sorted order $<a_1',a_2',\ldots\ldots a_n'>$ such that $a_1'\leq a_2'\leq\ldots\leq a_n'$.

Sample Input

6
712
-45
456765
0
-8907
4566

6. Radix Sort

Implement the radix sort algorithm, to sort *n* non-negative hexadecimal numbers. The numbers can range from 0000 0000 to FFFF FFFF. The numbers must be sorted in hexadecimal format itself and not after conversion to integer format.

Input: An array on *n* hexadecimal numbers, $<a_1,a_2,\ldots\ldots a_n>$.
Output: The n hexadecimal numbers in sorted order $<a_1',a_2',\ldots\ldots a_n'>$ such that $a_1'\leq a_2'\leq\ldots\leq a_n'$.

Sample Input

```
6
BEE
CAFEF1FA
00000000
7CD
101248
A5A5A5A5
```

7. Merge Sort

Implement the *merge sort* algorithm to sort *n* integers.

Input: An array of n integers, $<a_1,a_2,...,a_n>$
Output: The *n* integers in sorted order $<a_1',a_2',\ldots\ldots a_n'>$ such that $a_1' \leq a_2' \leq \ldots \leq a_n'$.

Sample Input

```
6
712
-45
456765
0
-8907
4566
```

8. Heap Sort

Given a list of ***n* words**, implement the heap sort algorithm, to sort them in l*exicographical order*. The words contain only upper case English letters (**A** – **Z**).

Input: An array of ***n*** words consisting of upper case English letters (**A**-**Z**).
Output: The n words sorted in lexicographical order.

Sample Input

```
6
NITC
SORTING
DATASTRUCTURES
SORT
SELECT
INSERT
```