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

Submission deadline (on or before):
24th January 2016, 10:00:00 PM (for both Main and Advanced batches)

Policies for Submission and Evaluation

You must submit your assignment in the moodle (Eduserver) course page, on or before the submission deadline. Also, ensure that your programs in the assignment must compile and execute without errors in Athena server. During evaluation your uploaded programs will be checked in Athena server only. Failure to execute programs in the assignment without compilation errors may lead to zero marks for that program.

Your submission will also be tested for plagiarism, by automated tools. In case your code fails to pass the test, you will be straight away awarded zero marks for this assignment and considered by the examiner for awarding F grade in the course. Detection of ANY malpractice regarding the lab course will also lead to awarding an F grade.

Naming Conventions for Submission

Submit a single ZIP (.zip) file (do not submit in any other archived formats like .rar or .tar.gz). The name of this file must be ASSG<NUMBER>_<ROLLNO>_<FIRST-NAME>.zip (For example: ASSG2_BxxyyyyCS_LAXMAN.zip). DO NOT add any other files (like temporary files, input files, etc.) except your source code, into the zip archive.

The source codes must be named as ASSG<NUMBER>_<ROLLNO>_<FIRST-NAME>_<PROGRAM-NUMBER>.<extension> (For example: ASSG2_BxxyyyyCS_ LAXMAN_1.c). If there is a part *a* and a part *b* for a particular question, then name the source files for each part separately as in ASSG2_BxxyyyyCS_LAXMAN_1b.c.

If you do not conform to the above naming conventions, your submission might not be recognized by some automated tools, and hence will lead to a score of 0 for the submission. So, make sure that you follow the naming conventions.

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.

Assignment Questions

General Instructions for all the questions:

Invalid input should be detected and suitable error messages should be generated. Sample inputs are just indicative.

1) Write a program which reads a list of integers sorted in strictly increasing order and an integer *k*; and uses binary search to find *k* in the list. Implement the following versions of binary search.

a) Iterative version

b) Recursive version

Input format:

The first line contains an integer *n*, indicating the size of the list.

The next line contains *n* space separated integers.

The next line contains an integer *k*.

Output Format:

If *k* is present, then print the index of *k* in the list.

If *k* is not present, then print the message "NOT FOUND"

Example 1:

Input:

```
8
12 15 25 35 39 42 45 75
35
```

Output:

```
4
```

Example 2:

Input:

```
8
12 15 25 35 39 42 45 75
55
```

Output:

```
NOT FOUND
```

2) Write a program that reads an *n×m* integer matrix *M* and an integer *k*, and checks whether *k* is present in *M* or not. If *k* is present in *M*, then print the indices of all its occurrences, otherwise print the message "NOT FOUND".

Input format:

The first line contains two space separated integers *n* and *m*, the number of rows and columns of the matrix.

The next n lines contain m space separated integers, each line representing the elements in a row of the matrix.

The last line contains an integer *k*.

Output Format:

If *k* occurs *t* times in the input, then your output should have *t* lines, with i^{th} line containing two space separated integers x_i and y_i, where x_i and y_i are the row and column numbers of i^{th} occurrence of *k,* respectively in the matrix. Here, $1 <= i <= t$.

If *k* is not present in the matrix, then print the message "NOT FOUND".

Example 1:

Input:

```
4 4
457 639 231 593
827 29  166 141
54  932 61  845
61  41  735 820
61
```

Output :

```
3 3
4 1
```

Example 2:

Input:

```
2 4
457 639 231 593
827 29  166 141
185
```

Output :

```
NOT FOUND
```

3) You are given an array STUDENT_LIST of structures STUDENT with fields: ROLL_NO, NAME and MARK. (ROLL_NO is a string of length 9 of the form B140123CS, NAME is a string and MARK is an integer in the range 0-100)

a) Implement the recursive Quick Sort algorithm to sort STUDENT_LIST in descending order with respect to MARK field in the STUDENT record.
b) Implement the Radix Sort algorithm to sort STUDENT_LIST in ascending order with respect to ROLL_NO field in the STUDENT record.

Input Format*:*

The first line contains a positive integer *n* indicating the length of the array.
Each of the next *n* lines contain three space separated strings: ROLL_NO, NAME and MARK

Output Format:

Question 3a) Print the sorted STUDENT records in the descending order of MARK field.
Question 3b) Print the sorted STUDENT records in the ascending order of ROLL_NO field.

Sample input:

```
3
B140121CS Bob 75
B131051EC John 83
B140023EC Alice 54
```

4) Implement the bubble sort algorithm, to sort *n real numbers* in ascending order.

Input format:

The first line contains a positive integer *n* indicating the length of input array.
The next line contains *n* space separated real numbers.

Output Format:

The output contains *n* space separated real numbers of the sorted array.

Sample input:

```
8
623.43 82.412 534.612 137.6 -733.216 28.16 363.532 8.34423
```

5) Implement the insertion sort algorithm, to sort an array of unsigned integers in the ascending order.

Input format:

The first line contains a positive integer *n* indicating the length of the array.
The next line contains *n* space separated unsigned integers of the input array.

Output Format:

The output contains *n* space separated sorted unsigned integers.

Sample input:

```
8
854 73 342 882 214 74 184 79
```

6) Implement the Selection sort algorithm, to sort an array of strings in the lexicographical order.

Input format:

The first line contains a positive integer *n* indicating the number of strings.
The next line contains *n* space separated strings, where each string contains only lowercase letters [a-z] and has length at most 10.

Output Format:

The output contains *n* strings, sorted lexicographically.

Sample input:

```
9
one picture is worth more than ten thousand words
```

Sample output:

```
is more one picture ten than thousand words worth
```

7) Implement the Heap sort algorithm, to sort an array of integers.

Input format:

The first line contains an integer *n* indicating the length of input array.
The next line contains *n* space separated integers.

Output Format:

The output contains *n* space separated integers in the sorted order.

Sample input:

```
8
-854 73 -342 -882 214 -74 184 79
```

8) You are given a file NUMBERS containing at most 10000 integers, with each integer on a new line. Read the contents of this file NUMBERS, use merge sort to sort them recursively and output the result into another file.

Input Format:

The input file has at most 10000 lines, each containing an integer.

Output Format:

The output file should contain the integers in ascending order, with each integer printed on a new line.

****