Birla Institute of Technology & Science, Pilani
2nd Semester 2016-17 - CS F211 – Data Structures and Algorithms

---

Lab 6 (Evaluation 2): 4th Mar, 2017

Time: 170 minutes **Marks: 15+ 15 = 30**

Instructions:

- *This test consists of two problems (Problem 1 and Problem 2) specified in two different files.*
- All input expressions should be read from stdin (scanf) and output should be printed on stdout (printf).
- For first 150 minutes, only a subset of test cases will be visible to students after submitting the code on the portal. Only in last 20 minutes, all test cases will be made visible.
- At the end of 170 minute period, the online system will stop evaluating the submissions but it will accept it for additional 10 minutes. At the end of 180 minute period, it will stop accepting the submissions.
- Only the last submission by the student for each problem will be considered for evaluation, irrespective of earlier correct submission.
- Assuming that a problem contains M marks, in case of (Run-error/Compiler-error/Timelimit-error), evaluation will be done for M/2 marks only.
- Total marks of each problem contains some marks for modularity and proper structuring of code.
- All submitted source code will be later checked manually by the instructor and final marks will be awarded. Any case of plagiarism and/or hard coding of test cases will fetch 0 marks for the problem/evaluation component.
- Make sure to return 0 from the main() function in case of normal termination.

Problem 1 of 2

Expected Time: 85 minutes **Marks: 15**

Problem Statement

In this problem we will implement a string sorting algorithm known as "**Least Significant Digit (LSD) Radix Sort**" to sort an array of **fixed-length** character strings. For sake of simplicity we will assume that the input strings contain only small case alphabetic characters ([a..z]). Note that your implementation should not use any of the string comparison functions provided as part of C library.

Description of LSD Radix Sort

The LSD Radix Sort algorithm uses a simple technique called "**Key Indexed Counting**". The Key Indexed Counting is another sorting algorithm to sort an array A[] of N integers which can take the values between 0 and R-1. In case of alphabetic characters (instead of integers), the range $[a..z]$ can be considered as numeric range $[0..25]$.

The Key Indexed Counting algorithm works as follows:

Input: sort an array A[] of N characters in the range [a..z]

1. Count frequencies of each letter using key (its numeric value) as index.
2. Compute frequency cumulates which specify destinations.
3. Access cumulates using key as index to move items in a temporary array.
4. Copy temporary array back into original array.

Example (Key Indexed Counting):

i	A[i]
0	d
1	a
2	c
3	f
4	f
5	b
6	d
7	b
8	f
9	b
10	e
11	a

Initial array

char	frequency
a	2
b	3
c	1
d	2
e	1
f	3

Character Frequencies

char	frequency cumulates
a	2
b	5
c	6
d	8
e	9
f	12

Cumulative character frequencies

i	A[i]
0	a
1	a
2	b
3	b
4	b
5	c
6	d
7	d
8	e
9	f
10	f
11	f

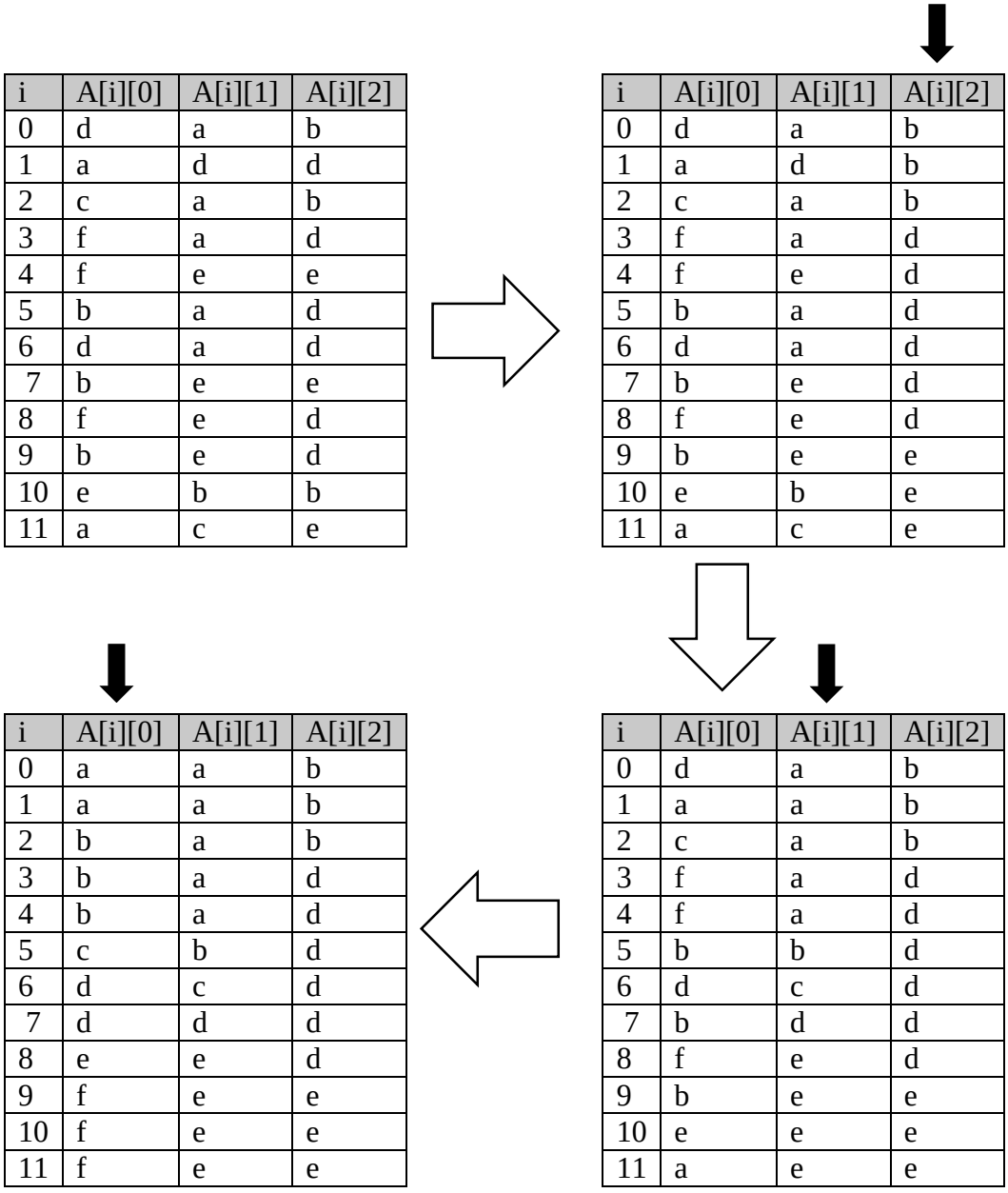
Final array

Now, we will implement the LSD Radix sort algorithm (that uses the concept of key indexed counting), to sort an array of fixed length character strings.

1. Consider the characters of a string as elements of a two dimensional array, such that
   a. Array of strings : char A[][]
   b. The i^{th} string : A[i]
   c. The d^{th} character of the i^{th} string : A[i][d]
   d. Strings to be sorted : A[lo], ……., A[hi]

2. Write a function $keyindex(char\ \boldsymbol{A}[][\ ], int\ \boldsymbol{low},\ int\ \boldsymbol{high}, int\ \boldsymbol{d})$ which will perform the key indexed counting (as described above) on the array of elements belonging to the range A[low][d] to A[high][d].
3. Write a function $lsdRadixSort(char\ \boldsymbol{A}[][\ ], int\ \boldsymbol{low},\ int\ \boldsymbol{high})$ to sort N fixed size (size = W) strings stored in array A as form of character pointers. To implement this, perform key-indexed counting repeatedly from right ($keyindex(A,\ 0,\ N-1, W-1)$) to left ($keyindex(A,\ 0,\ N-1, 0)$.

Example (LSD radix sort using Key Indexed Counting):



i	A[i][0]	A[i][1]	A[i][2]
0	d	a	b
1	a	d	d
2	c	a	b
3	f	a	d
4	f	e	e
5	b	a	d
6	d	a	d
7	b	e	e
8	f	e	d
9	b	e	d
10	e	b	b
11	a	c	e

i	A[i][0]	A[i][1]	A[i][2]
0	d	a	b
1	a	d	b
2	c	a	b
3	f	a	d
4	f	e	d
5	b	a	d
6	d	a	d
7	b	e	d
8	f	e	d
9	b	e	e
10	e	b	e
11	a	c	e

i	A[i][0]	A[i][1]	A[i][2]
0	d	a	b
1	a	a	b
2	c	a	b
3	f	a	d
4	f	a	d
5	b	b	d
6	d	c	d
7	b	d	d
8	f	e	d
9	b	e	e
10	e	e	e
11	a	e	e

i	A[i][0]	A[i][1]	A[i][2]
0	a	a	b
1	a	a	b
2	b	a	b
3	b	a	d
4	b	a	d
5	c	b	d
6	d	c	d
7	d	d	d
8	e	e	d
9	f	e	e
10	f	e	e
11	f	e	e

Note: *For radix sorting to work the individual sorts (i.e. sorting on one digit) must be stable.*

Input format

Each line will start with a one of the following key values (1, 2, 3, and -1). Each key corresponds to a function and has a pattern. Implement following function according to given pattern and write a driver function which can call the functions according to given key value.

Key	**Function to call**	**Format**	**Description**
1	$readStrings()$	1 N W	Indicates that next M lines will contain data to be read. Each line will contain a string of alphabetic characters (only small letters) of size W. Store the strings in an array (say A)
2	$keyindex()$	2 d	Perform Key Index Counting to sort the characters stored in A at **d**th index. Print the array of sorted characters in single character per row format i.e. print characters at **d**th index only. If it is being called from driver function, then perform it on a copy of original input. Counting for **d** starts from 0.
3	$lsdRadixSort()$	3	Perform $lsdRadixSort()$ on A and print each string of sorted array A in a new line.
-1			stop the program

Test Case

Input	**Output**
1 10 6 xayjjh vnbaew fhljtc cynfhy jjlgkk urmpeo jkaprm koamno umadwm hwzbom 2 3 2 4 3 -1	a b d f g j j m p p e e h j k n o r t w

	cynfhy fhljtc hwzbom jjlgkk jkaprm koamno umadwm urmpeo vnbaew xayjjh