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 2 of 2

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

Problem Statement

In this problem, you have to implement a sorting algorithm known as **"Three-Way Radix-Partition Sort"** on a list of Martian DNA strings:

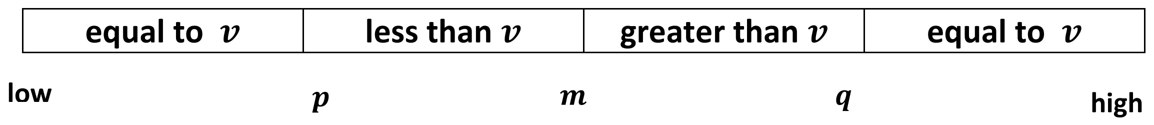
Each Martian DNA string is a fixed-size sequence of characters where each character is one of X, Y, or Z. Assume X < Y < Z.

This algorithm combines the concepts of partitioning (from quicksort) and radix sort. To be more specific, the algorithm applies partitioning position-by-position in a string. It is important that your implementation should not use any of the string comparison functions provided as part of C library.

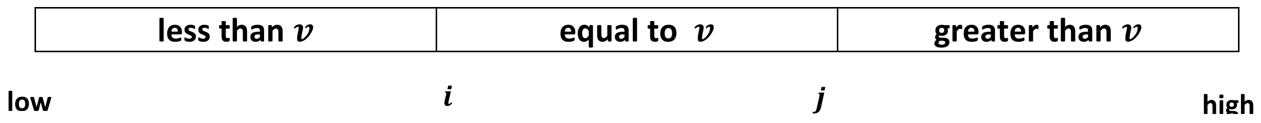
3-Way Partitioning

An efficient way of doing 3-way partitioning is as follows:

1. First partition the list in *<u>four parts</u>*, given pivot value v :
   - **Part 1:** no larger elements (than v) to the left of m
   - **Part 2:** no smaller elements (than v) to the right of m
   - **Part 3:** equal elements (to v) to the left of p
   - **Part 4:** equal elements (to v) to the right of q



2. After partitioning in the above manner, move elements with key equaling the pivot to the center so that:
   - **Part 1:** Elements between i and j equal to partition element v
   - **Part 2:** No larger elements to the left of i
   - **Part 3:** No smaller elements to the right of j



Note: These two steps can be achieved while maintaining stability i.e.
if, before partitioning:

```
     for some i<j, Ls[i]=v1 and Ls[j]=v2 where
v1.key=v2.key
```

then after partitioning:

```
     v1=Ls[i1] and v2=Ls[j1] for some i1 and j1 such that
i1<j1.
```

End of Note.

Implementation:

a) intPair part3way3keys(char *Ls[], int lo, int hi, int d)

such that **Ls[lo..hi]** is split into 3 parts **Ls[lo..i]**, **Ls[i+1..j]**, **Ls[j+1..hi]** containing all strings with **d**th character as **X**, **Y**, and **Z** respectively (i.e. use **dth** character as key and **Y** as the pivot). This procedure should return a pair **(i,j)** of indices that form the boundaries as described in the previous statement. Counting for **d** starts from 0.

b) radixPart3waySort(char *Ls[] , int size)

Use procedure ***part3way3keys()*** to design the *Three-Way Radix-Partition Sort* algorithm:

```
    repeatedly, sort the input list of Martian DNA
strings on the basis of one of the characters as a key,
at a time. Select keys in the order in which it will
produce correct sorted result.
```

[**Hint:** part3way3keys() *is a stable sorting procedure for any list of elements with only three distinct keys.* **End of Note**.]

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 N lines will contain data to be read. Each line will contain Martian DNA string of W letters. Store these strings as an array of character pointers (**say char *Ls[]**).
2	*part3way3keys()*	2 d	Call $part3way3keys()$ with **d**th character as key on *Ls[] with print modified *Ls[], each string on a new line. Counting for ***d*** starts from 0. If it is being called from driver function, then perform it on a copy of original input.
3	radixPart3waySort()	3	Call radixPart3waySort() on *Ls[] to sort the strings and print every string of sorted array in a new line.
-1			stop the program

Test Case

Input	Output
1 10 4 ZXYX YZXX ZXXY ZYZY YXXX XYYX XXZY YXXY ZXYZ XZYZ 3 -1	XXZY XYYX XZYZ YXXX YXXY YZXX ZXXY ZXYX ZXYZ ZYZY