Birla Institute of Technology & Science - Pilani, Hyderabad Campus
Second Semester 2014-2015
CS F211 / IS F211 : Data Structures and Algorithms
Test 2

Type: Closed **Time: 60 mins** **Max Marks: 60** **Date: 07.04.2015**

All parts of the same question should be answered together.

1.a. Describe, in pseudo code , a link-hopping method for finding the middle node of a doubly linked list with header and trailer sentinels, and an odd number of real nodes between them. (Note: This method must only use link hopping; it cannot use a counter.) What is the running time of this method? [6 Marks]

1.b. (i). The median of a set of n values is the $(n/2)^{th}$ smallest value. Suppose quicksort always pivoted on the median of the current sub-array. Derive recursive relation for the number of comparisons that Quick Sort would make in worst case?
(ii) Suppose quicksort were always to pivot on the $(n/3)^{rd}$ smallest value of the current sub-array. Derive recursive relation for the number of comparisons that Quick Sort would make in worst case? [2 + 3 Marks]

1.c. An ordered list is a linked list where all elements are arranged in ascending order. If there are 'n' elements in an ordered list and if a new element is to be inserted then it is put up in the list in its correct position. For example if a new element is to be inserted and say it happened to be the 8th largest element of 'n+1'elements then it is placed between 7th and 9th largest elements. Design an algorithm to achieve this functionality and find the average case complexity of this algorithm. [7 Marks]

2.a. Let H be a heap storing 30 unique integer keys. List all the levels of the heap H that can have the 7th smallest key. Assume that smaller keys have higher priorities. [4 Marks]

2.b. Give an O(n log k)-time algorithm that merges k sorted lists with a total of n elements into one sorted list. [8 Marks]
Note: You should use a heap to speed up the elementary O(kn)- time algorithm. Any other solution will not be considered for evaluation.

2.c. Give an efficient algorithm to find the second-largest key among n keys. You can do better than $2n - 3$ comparisons. [10 Marks]
Hint: Think of solving this problem with the help of complete binary tree having 'n' leaves.

3.a. Develop an algorithm that computes the kth smallest element of a set of n distinct integers in O(n + k log n). You should make use of the data structure, Heap, in your algorithm. [8 Marks]

3.b. Describe how to implement the stack ADT using two queues. What is the running time of the push and pop methods in this case? [3 Marks]

3.b. Suppose 'n' comparable elements are put up in an array then prove that max-heap can be built in O(n). [9 Marks]
Note: You have to prove all necessary lemmas / theorems to prove the complexity of this algorithm.