DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING
CONCORDIA UNIVERSITY
COMP 352: Data Structure and Algorithms
Summer 2019
THEORY ASSIGNMENT 4
Due date and time: Thursday, June 20th, by 11:59 pm

Written Questions (110 marks)

Q1. Draw the 13-entry hash table that results from using the hash function *h(k) = (3k - 5) mod 13* to hash the keys 31, 45, 14, 89, 24, 95, 12, 38, 27, 16, and 25, assuming that collisions are handled in the following ways:

a) linear probing
b) double hashing using the secondary hash function *h'(k) = 7 – (k mod 7)*.

Q2. Show (diagrammatically in the form of a divide-and-conquer tree, in the same format shown in your text book and slides) all the steps involved in the **in-place quick sort** on the following input sequence: 22 72 38 48 13 14 93 69 45 58 13 81 79. You should sort the sequence in the non-decreasing order. Clearly show the pivots chosen at each node. (Note: Though in practice the pivot for each input is chosen at random, you may choose pivots here of your choice which make the tree balanced, i.e. only choose *good* pivots).

Q3. Answer the following questions:

a) Explain why Merge sort is the most suited for very large inputs (that do not fit inside memory) while quick sort and heap sort are not as suited. Note that these three sorting techniques have comparable time complexities.
b) Can Merge sort be performed in place? Explain your understanding.

Q4. Answer the following questions:

a) Consider the following sequence of keys inserted into an initially empty AVL tree T in this specific order: 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1. Draw the AVL tree T after each insertion.
b) Is an AVL tree for a specific set of inputs unique? If your answer is yes, then explain why it is so. If your answer is no then demonstrate using an example.

Q5. Answer the following questions:

a) Draw an AVL tree of height 3 that has the minimum number of internal nodes.
b) Draw an AVL tree of height 5 that has the minimum number of internal nodes.
c) In general, briefly explain how will you draw an AVL tree of height *h* that has the minimum number of internal nodes.

Q6. Describe an efficient algorithm for computing the height of a given AVL tree. Your algorithm should run in time *O(log n)* on an AVL tree of size *n*. In the pseudocode, use the following terminology: T.left, T.right, and T.parent indicate the left child, right child, and parent of a node T and T.balance indicates its balance factor (-1, 0, or 1).

For example if T is the root we have T.parent=nil and if T is a leaf we have T.left and T.right equal to nil. The input is the root of the AVL tree. Justify correctness of the algorithm and provide a brief justification of the runtime.

Q7. Given a balanced binary search tree that somehow allows duplicates populated by a sequence S of n elements on which a total order relation is defined, describe an efficient algorithm for determining whether there are two equal elements in S. What is the running time of your algorithm?

Q8. A graph $G(V, E)$ is *bi-partite* if $\exists A, B, A \cap B = \emptyset, A \cup B = V\ s.t. \forall (x, y) \in E, (x \in A\ \wedge y \in B) \vee (x \in B\ \wedge y \in A)$. Prove that a tree is always bi-partite.

Q9. Prove that a graph is bi-partite if and only if it has no odd cycles.

Q10. Given a graph $G(V, E)$ write the pseudocode of an algorithm that computes a bi-partite partitioning of the graph if one exists or returns false if none exists.

Q11. A *complete* graph is a graph where all vertices are connected to all other vertices. A *Hamiltonian path* is a simple path that contains all vertices in the graph. Show that any complete graph with 3 or more vertices has a Hamiltonian path. How many Hamiltonian paths does a complete graph with n vertices has? Justify your answer.