Department of EECS
York University

Term: Winter 2019
Instructor: Andy Mirzaian

EECS2011: Fundamentals of Data Structures
Sections M and Z
Assignment 3
Due: 10 pm, Friday, March 15, 2019

- **Read the course FAQ on how to submit assignments electronically.**
- **Print your name, eecs account, and student ID # on top of every file you submit.**
- **You may give your algorithms in either java code or pseudo-code.**

Problem 1: [15%] ([GTG] Exercise R-8.11, page 350)

Draw an arithmetic expression tree that has four external nodes, storing the numbers 1, 5, 6, and 7 (with each number stored in a distinct external node, but not necessarily in this order), and has three internal nodes, each storing an operator from the set { +, –, *, / }, so that the value of the root is 21. Any of these operators may be used more than once, and they are real (not integer) operators, i.e., they may operate on and return fractions.

Problem 2: [35%] ([GTG] Exercise C-8.57, page 355)

Let T be a given binary tree with n nodes. The **distance** between two nodes p and q in T is the number of edges along the unique simple path between p and q, i.e., $d_p + d_q - 2d_a$, where a is the lowest common ancestor (LCA) of p and q, and d_x denotes the depth of node x in T. The **diameter** of T is the maximum distance between two nodes in T (i.e., the distance between the farthest pair of nodes in T). Give an efficient algorithm for computing the diameter of T and analyze its running time.

[*Note: Each node of T stores an element and a pair of references to its left and right children. Other than that, nodes of T do not have any additional space to store extra information, not even their depths. Also, you should not attempt to create a clone of T with extra spaces in the clone nodes. Hint: use a post-order traversal of T with strengthened post-condition. As an example, see how we use a composite Result type as the return type in the EulerTour algorithm , LS8, pp: 20-21.*]

Problem 3: [20%] ([GTG] Exercise C-9.38, page 398)

Tamarindo Airlines wants to give a first-class upgrade coupon to the top $\log n$ of their frequent flyers, based on the number of miles accumulated, where n is the number of the airlines' frequent flyers. The algorithm they currently use, which runs in $O(n \log n)$ time, sorts the flyers by the number of miles flown and then scans the sorted list to pick the top $\log n$ flyers. They have hired you as their chief software engineer.

Give an algorithm that identifies the top $\log n$ flyers in $O(n)$ time.

Problem 4: [30%] Dynamic Median Finder (DMF):

We want to design a DMF ADT that maintains a collection of comparable elements and supports the following operations on the collection:

- *insert(e):* inserts a given element e in $O(\log n)$ time,
- *getMed():* returns the median in $O(1)$ time,
- *removeMed():* removes and returns the median in $O(\log n)$ time,

where n denotes the current number of elements in the collection.

Give an implementation of the DMF ADT using two heaps as the only instance variables.

[*The median of a collection of n elements is the $\lceil n/2 \rceil^{th}$ smallest element (ties broken arbitrarily). For instance, the median of $\langle 4, 9, 1 \rangle$ is 4, the median of $\langle 9, 3, 3 \rangle$ is 3, the median of $\langle 9, 9, 1, 2 \rangle$ is 2, and the median of $\langle 17, -4, 13, -7, 13, 15, 5, 2 \rangle$ is 5.*]

What files to submit:

- **a3sol.pdf :** Describe solutions to the problems with detailed explanations and illustrations of your algorithmic design ideas, all your pseudo-codes, convincing proofs of correctness and time complexity analysis.
- Submitting java source codes are optional.