CIS 313, Intermediate Data Structures
Fall 2016, University of Oregon

Due: 3:30pm October 21th, 2016
Date: October 7, 2016

Assignment 2

Grades : each assignment is 5% in the final score and there is a 2% bonus in each assignment for bonus problems. We use the following scale for the simplicity of grading: the total score is 50 (additional 20 for bonus problems) for each assignment.

Problem 1 [Full Score: 15]. C-2.2 (on page 133). We talked about how to implement a queue by two stacks. Recall and formulate the idea. Then prove that the amortized complexity of dequeue() and enqueue() is $O(1)$.

(You are only required to do the analysis using one method. You are free to choose from the accounting method and the potential function method. You are encouraged to try both as your own exercise.)

Problem 2 [Full Score: 20]. **Minqueues**. Consider an extension of the queue abstract data type, called a minqueue, which supports operations ENQUEUE, DEQUEUE, and FINDMIN. Assume that the elements to be stored are integers (or any other totally ordered data type). FINDMIN simply returns the smallest element in the minqueue, but does not remove it. Using a standard implementation of a queue, the FINDMIN operation takes $O(n)$ time in the worst case. However, consider a more clever data structure for this abstract data type which works as follows: There are two regular queues used. The first queue is called the "real queue" and the second queue is called the "helper queue". When the ENQUEUE(x) operation is performed, the element x is enqueued in the regular way on the real queue. The element x is also enqueued at the end of the helper queue. However, if the element y immediately "in front" of x on the helper queue is larger than x, then y is removed from the helper queue. This process of x annihilating the element immediately in front of it is repeated until the element immediately in front of x is less than or equal to it or x is at the front of the helper queue.

(a)[4] Describe how the DEQUEUE and FINDMIN operation are implemented in this data structure.

(b)[6] Give the worst case running time for each of ENQUEUE, DEQUEUE, and FINDMIN using this data structure.

(c)[10] Give an argument using the potential method to show that any sequence of n ENQUEUE, DEQUEUE, and FINDMIN operations requires only $O(n)$ time. Or namely, the amortized complexity of these operations is $O(1)$.

Problem 3 [Full Score: 5]. Draw the (improper) binary tree whose inorder traversal is abcdefgh and whose postorder traversal is acbegfhd.

Problem 4 [Full Score: 10]. C-2.12 (on page 134).

Problem 5 [Bonus Problems: 20]. Assume there is an implementation of binary tree ADT such that one can access its root, the key stored in each node, and the left and the right child of each node with $O(1)$ complexity. Let T, given as input, be such an object with n nodes.

(1)[10] Describe an algorithm that checks whether T is a valid binary search tree. Analyze the worst-case complexity of your algorithm.

(2)[10] Assume T is a binary search tree and let k be another input. Describe an algorithm that finds one of the closest-to-k keys in the binary tree T. Analyze the worst-case complexity of your algorithm. (Assume all the keys are integers and the distance between two keys is the absolute value of their difference.)