Department of Computer Science **University of San Francisco**

Computer Science 673
Fall 2016
Homework 7: Amortized Analysis
Due Friday, October 21st

All problem / exercise numbers are from the **3nd edition** of Introduction to Algorithms (the 1st and 2nd editions are different!)

1. (6 points) Show how to implement a queue with two ordinary stacks so that the amortized cost of each ENQUEUE and each DEQUEUE is O(1). That is, using only 2 stacks (that only support the operations PUSH, POP, and EMPTY) to store your data, give pseudocode for ENQUEUE and DEQUEUE. Use the potential method to show that both of your operations require amortized time $O(1)$.

2. (6 points) Consider a data structure that consists of a standard sorted array, which takes time $O(n)$ to insert into and delete from. Create a potential function for this data structure that results in amortized time $O(n)$ for insertion and $O(1)$ for deletion. Prove the amortized running times. Note that you are not supposed to *change* the operations of insert and delete, just compute amortized running times.

3. (10 points) Problem 17-2 Making Binary Search Dynamic

4. (6 points) Give pseudocode for a function that takes as input a B-Tree T and a key k, and returns the successor of k (or nil, if k is the largest key in the tree). Your code may assume that k appears in the tree, and that all keys are unique. Furthermore, assume that each node of the tree has fields numkeys, isLeaf, keys[], and children[]. So, a B-Tree node N containing the keys 4, 5, 7 which is not a leaf would have:

   - N.numkeys == 3
   - N.isLeaf == false
   - N.keys[1] == 4, N.keys[2] == 5, N.keys[3] == 7
   - children nodes in N.children[1], N.children[2], N.children[3], N.children[4]

   successor(T, k)