CSE 332 Data Abstractions, Winter 2014
Homework 4

Due: **Wednesday, February 5** at the BEGINNING of lecture. Your work should be readable as well as correct. You should refer to the written homework guidelines on the course website for a reminder about what is acceptable pseudocode.

Problem 1: B-Tree Insertion

Show the result of inserting 38, 11, 17, 3, 42, 51, 7, 14 & 15 in that order into an initially empty B tree with M = 3 and L = 2. (Recall the text, lecture, and this problem call a B tree what many call a B+ tree.) Show the tree after each insertion, clearly labeling which tree is which. In an actual implementation, there is flexibility in how insertion overflow is handled. However, in this problem, follow these three guidelines:

- Always use splitting (not adoption).
- Split leaf nodes by keeping the smallest 2 elements in the original node and putting the 1 largest element in the new node.
- Split internal nodes by keeping the 2 children with the smaller values attached to the original node and attach the 2 children with the larger values to the new node.

Problem 2: B Tree Predecessor

In this problem, assume every B tree node has a reference to its parent (except for the root node) as well as its children (except for leaves), so that it is easy to move up or down" the tree.
Suppose you have a direct reference to the leaf holding a data item with a key k.

a) Describe in English an algorithm for finding the predecessor of k in the B tree (or determining that k is the minimum element and therefore has no predecessor).
b) Give the worst-case number of total nodes accessed by your algorithm in terms of the tree height h. Describe briefly a worst-case situation.
c) Give the best-case number of total nodes accessed by your algorithm. Explain briefly why most keys would exhibit the best-case behavior.

Problem 3: Textbook exercise

Weiss, Chapter 5, exercise 5.1. (Page 218 in the 3rd edition, page 194 in the 2nd edition.)

Problem 4: Deletion in Hashing

In this problem you will think about how lazy deletion is handled in open addressing hash tables.
(a) Suppose a hash table is accessed by open addressing and contains a cell X marked as "deleted". Suppose that the next successful find hits and moves past cell X and finds the key in cell Y. Suppose we move the found key to cell X, mark cell X as "active" and mark cell Y as "open". Suppose this policy is used for every find. Would you expect this to work better or worse compared to not modifying the table? Explain your answer.
(b) Suppose that **instead** of marking cell Y as "open" in the previous question, you mark it as "deleted" (it contains no value, but we treat it as a collision). Suppose this policy is used for every find. Would you expect this to work better or worse compared to **not modifying** the table? Explain your answer.