Homework 4

CSE332, Due Feb. 3rd at the beginning of lecture.
Note that throughout this homework we use 'B Tree' to mean 'B+ Tree'; these conform to the rules described in lecture.

Problem 1. Practice with B Trees

For the following questions about B trees, show the tree after each insert or delete.

a) Show the result of inserting 12, 10, 15, 4, 1, 17, 3, 13, and 8 into an initially empty B tree with M = 3 and L = 2. To maintain consistency in answers, please follow the following rules:
   - You should split nodes whenever there is an overflow due to insertion; that is, do not use adoption.
   - When splitting a leaf node due to insertion overflow, keep half (rounded up) in the left node and half (rounded down) in the right.

b) Now show the result of deleting 12, 13, and 15.

Problem 2. B Tree Predecessor

In this problem, assume that 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. Practice with Hashing

This will give you a chance to get some practice with hash tables, which you will be using in the current project.

a) For each of the following types of hash-tables, insert the following values in order: 4371, 1323, 6173, 4199, 4344, 9679 and 1989. Assume the table size is 10 and that the primary hash function is h(k) = k % 10. You do not need to resize the tables. If an element cannot be successfully inserted, state why. You need only show the final table.

   i. Separate chaining hash table
   ii. Hash table using linear probing

iii. Hash table using quadratic probing
iv. Hash table with a secondary hash function of h2(k) = 7 – (k % 7)

b) Show the result of rehashing each of your results from part (a). Choose the new table size to be 19, which is prime and roughly twice as big. When rehashing, insert in the order in which they are present in your table from part (a) (this is how rehashing is usually done anyway). If any items were not successfully inserted in part (a), they should be inserted after the rehash, in the order of their failures.

c) 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.

d) 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.