CSE332 Data Abstractions, Summer 2012
Homework 4

Due: **Thursday, July 19, 2012** by the end of the last quiz section that day. You work should be readable as well as correct. You should refer to the written homework guidelines on the course website.

This assignment has **3** questions.

For all problems, it may help with partial credit to show other intermediate steps.

Problem 1. 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 (internal and leaf 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 2. 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 and throughout your programming careers.

For parts (a)-(d), you will be inserting into a hashtable using the indicated collision resolution technique. The following are the values in the order you will insert themm: 4371, 1323, 6173, 4199, 4344, 9679 and 1989. Assume the table size is 10 and that the primary hash function is $h(k) = k\%10$.

(a) Show the final hash table when separate chaining is used

(b) Show the final hash table when linear probing is used

(c) Show the final hash table when qradratic probing is used

(d) Show the final hash table when double hashing is used with $g(k) = 7 - (k\%7)$

(e) Show the result of rehashing each of your results from part (a)-(d). 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)-(d) (this is how rehashing is usually done anyway). If any items were not successfully inserted earlier, they should be inserted after the rehash, in the order of their failures.

Problem 3. Deletion in Hashing
This probelm is to get you to think about how lazy deletion is handled in hash tables using open addressing.

(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 .