COS 226 Fall 2009 HW03 200 + 40★ pts.; 2 pages. Due 2009-10-29 2:05 p.m.

Name: ______________________________

Please refer to the previous assignment for important instructions on the allowable use of resources and the requirements for electronic and hardcopy submission. Please monitor and use the class newsgroup for questions, clarifications, hints, and other discussions.

The questions in this assignment are very similar to those in the previous one, except that they refer to *level-wise* k-search-trees instead of k-search-trees in general, as explained in Question 4 below.

1. (1 pt.) Write your name in the space provided above.

2. (1 pt.) The pen-and-paper questions in this assignment need to be answered on separate sheets of paper that must be neatly attached to your hardcopy submission with a single staple. Note the number of additional pages attached here: ____

3. (1 pt.) Package and submit your solutions to the programming questions via `http://cs.umaine.edu/~chaw/u/`. *After* submitting your work, fill in the following:
   File name: ______________________
   Size, in bytes: ______________________
   MD5 checksum: ______________________
   Timestamp: ______________________

4. (22 pts.) Recall the definition of a *k-search-tree* from the previous assignment. Also recall that the *level* of the root of a tree is 0, and the level of every other node is one greater than the level of its parent. Thus a tree of depth d has $d+1$ levels: $0, 1, \ldots, d$. Further, we say level s of a *k-search-tree* is *full* if it contains $(k-1)k^s$ keys (that is, k^s nodes with $k-1$ keys each). A *level-wise k-search-tree* is a k-search-tree that satisfies the following additional constraints:

   - It does not contain any node that has exactly one child.
   - If the depth of the tree is d then all the levels $0, 1, 2, \ldots, d-1$ are full.

   Depict all distinct labeled level-wise 4-search-trees with n nodes, for $n = 0, 1, \ldots, 7$, with labels drawn from the domain $\{1, 2, 3, 4, 5\}$. You may abbreviate the depiction of some trees *provided the result is unambiguous and very clear.*

5. (25 pts.) Describe an efficient algorithm for locating a given label in a given level-wise k-search-tree (or determining that it is absent from the tree). Provide both a clear English description and pseudocode. In addition, demonstrate a carefully selected illustrative example that highlights the main ideas used by your algorithm. Quantify the asymptotic running time of your algorithm as a function of n, the number of keys in the tree. Justify your answer.

6. (25 pts.) Describe an efficient algorithm for inserting a given label in a given level-wise k-search-tree (or determining that it is already in the tree). As in Question 5, provide both a clear English description and pseudocode, demonstrate an illustrative example, and quantify the asymptotic running time of your algorithm.

7. (25 pts.) Describe an efficient algorithm for removing a given label from a given level-wise k-search-tree (or determining that the tree does not contain that label). As in Question 5, provide both a clear English description and pseudocode, demonstrate an illustrative example, and quantify the asymptotic running time of your algorithm.

8. (90 pts.) Implement your algorithms of Questions 5, 6, and 7 as part of the simple record-manager application described in the previous assignment. (It may be convenient to simply modify your prior implementation to use the new algorithms.) Package and submit your source code as in the previous assignment, being sure to include adequate documentation. Poorly written or poorly documented code is likely to receive zero credit. In addition to the commands described in the previous assignment, implement the following:

command	actions
`d␣k`	If the record manager contains key-record pair (k, d) then that pair is deleted else there is no action. The output is an empty line in both cases.
`t␣1`	This command modifies the behavior of all commands issued after it, until a `t␣0` command is issued, in the following manner: After the previously described output for a command has been emitted, the time required to complete that command is reported, in milliseconds, as a decimal number on a line by itself.
`t␣0`	This command undoes the effect of the `t␣1` command. That is, as soon as this command is received, the application behaves as though the earlier `t␣1` command was not received.

9. (10 pts.) Conduct an experimental evaluation of each command of your record-manager implementation, following the guidelines in the previous assignment.

   Note the file names here:
   text results and explanation: ______________________________
   graphical results: ______________________________

10. (40 pts.) ⋆ Describe an efficient algorithm for generating all level-wise k-search-trees with n nodes, with labels drawn from L, where k, n, and L are provided as input. Provide an English description and pseudocode. Explain why the algorithm is correct. Determine and prove its running time. Implement the algorithm and conduct an experimental study of the running time. Package and submit your implementation as in earlier questions.