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 ask for clarifications in class and on the newsgroup.

Reminders: (1) Please ensure that everything you submit is neat and easy to comprehend; otherwise, it will be ignored, with zero credit. (2) The due date and time are strict; if you are even a second late, by gandalf's clock, you will get zero credit, so you are advised to not cut things too close.

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. (17 pts.) A *k-tree*, for $k > 1$, either is empty or consists of a node r, called its root, and a sequence of $c \leq k$ nonempty subtrees, each of which is a k-tree. The roots of these c subtrees are the *children* of r, and r is their *parent*.

   Depict all distinct (i.e., nonisomorphic) 4-trees with n nodes, for $n = 0, 1, 2, 3, 4, 5$.

5. (10 pts.) A *labeled k-tree* is a k-tree in which each node with $c > 1$ children is associated with a sequence of labels $l_1, l_2, \ldots, l_{c-1}$.

   Indicate the number of distinct labeled 4-trees with n nodes, for $n = 0, 1, 2, 3, 4, 5$, with labels drawn from the domain $\{1, 2, 3\}$. (A tree need not contain all the labels from the domain.) You do not need to depict the trees, but you must explain clearly how you arrive at the answers. (There is no credit for answers without proper explanations.)

6. (20 pts.) A *k-search-tree* is a labeled k-tree with the following additional constraints on each node n with $c > 1$ children, where we define $l_0 = -\infty$ and $l_c = \infty$ for convenience.

   - The labels are strictly increasing; that is, $l_i < l_{i+1}$ for $i = 1, 2, \ldots c - 1$.
   - All the labels in the subtree rooted at the ith child of n are greater than the label l_{i-1} and less than the label l_i, for $i = 1, 2, \ldots, c$.

Depict all distinct labeled 4-search-trees with n nodes, for $n = 0, 1, 2, 3, 4, 5$, with labels drawn from the domain $\{1, 2, 3, 4, 5\}$.

7. (10 pts.) Depict a k-search-tree in which some label l occurs in two distinct locations, or explain clearly why no such tree exists.

8. (10 pts.) Describe an efficient algorithm for locating a given label in a given k-search-tree (or determining that it is absent from the tree). Provide both a clear English description and pseudocode. Prove that the running time of your algorithm is $O(h)$ where h is the height of the k-search-tree.

9. (10 pts.) Describe an efficient algorithm for inserting a given label in a given k-search-tree (or determining that it is already in the tree). As in Question 8, provide both a clear English description and pseudocode, and prove that the running time of your algorithm is $O(h)$ where h is the height of the k-search-tree.

10. (10 pts.) Describe an efficient algorithm for removing a given label from a given k-search-tree (or determining that the tree does not contain that label). As in Question 8, provide both a clear English description and pseudocode, and prove that the running time of your algorithm is $O(h)$ where h is the height of the k-search-tree.

11. (10 pts.) Conduct an experimental evaluation of the three algorithms for the maximum contiguous subsequence sum problem described in the textbook.[1] Ensure that your results are meaningful, given the limited clock accuracy and other factors. In your electronic submission, include a *plain text ASCII* file that tabulates the running times for input sequences of varying lengths. Describe in that file how the input sequences are generated, and why that generation process is suitable. (Example: Do not exclusively use sequences that contain only positive or only negative numbers.) Explain any discrepancies between your results and those predicted by the textbook. Also include in your submission a *portable PDF* [sic] file that depicts the running time data in a suitable graphical form.

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

12. (90 pts.) Implement your algorithms of Questions 8, 9, and 10 as part of the simple record-manager application described below. 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.

    The record manager stores *key-data pairs* of the form (k, d) where k is a non-negative integer and d is floating point number. (You may assume that k and d fit in Java's `int` and `float` types.) All user interaction with the record manager is through a text-mode command language based on the standard-input/standard-output interface. The input consists of one command per line. The record manager reads and responds to each

[1] Mark Allen Weiss, *Data Structures and Problem Solving Using Java*, 3rd edition (Addison-Wesley, 2006), §5.3.

command in turn. Except for the `xp` command, the response to each command is also a single newline-terminated line. The syntax and semantics of the commands, and the desired outputs, are as follows. We use the symbol ␣ to denote a single space character.

command	actions
`c␣k`	Create a record manager that contains no records and that uses a k-search-tree with the given value of `k`. The output is an empty line (single newline character).
`s␣k␣d`	The pair (k, d) is stored in the record manager. If a pair of the form (k, d') already exists, then the new pair replaces it. The output is an empty line.
`e␣k`	If the record manager contains the key k then the output is `1` else the output is `0`.
`r␣k`	The data d associated with key k is retrieved from the record manager. The output is d. If the key k is absent from the record store, the output consists of an empty line.
`xs`	The output is the size of the tree (a single integer).
`xh`	The output is the height of the tree.
`xa`	The output is a list of the labels of the tree nodes in preorder. These labels (some with suffixes) are to be printed on a single line, with a single space separating adjacent entrees, with no additional punctuation such as commas or brackets.
`xb`	The output is similar to that of `xa` but the nodes are to be listed in postorder.
`xp`	The output consists of the following *two-dimensional text representation* of the current state of the tree used to store the records. This representation of a tree t contains one line for each node of t. The line representing a node $n \in t$ consists of n's label prefixed with $3d$ spaces, where d is the depth of n in t. Further, all the lines representing descendants of a node precede the line representing that node's right sibling, if any.

Ensure that (1) your program produces exactly the output described above, with no spurious text such as extra spaces or newlines, prompts, or other messages and (2) your program reads from standard input and writes to standard output, with no additional assumptions on either.

13. (10 pts.) Conduct an experimental evaluation of each command of your record-manager implementation, following the guidelines of Question 11.

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

14. (40 pts.) ⋆ Describe an efficient algorithm for generating all 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.