Name: ______________________________

This exam is open book, open notes, but there can be no sharing of any material. You can use the Internet, but only as a library. If you are not sure if something is allowed, please ask. If you use any material other than the textbook and your own prior work, you must prominently indicate the source in your answers.

Questions marked with ⋆ are optional and may be answered for extra credit. There are 7 questions (including one ⋆ question) on **??** pages. You have *60 minutes* to earn *60 non-⋆ points*. You may wish to use this correspondence to plan your time.

Unless otherwise indicated, use the textbook's definitions for all terms, such as trees, balancing conditions, and red-black tree insertions and rotations.

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

2. (14 pts.) For each statement below, indicate whether it is true or false by writing *True* or *False* in the blank next to it.

   (a) In any nonempty tree, there is exactly one node that has no parent. ________

   (b) In any nonempty tree, there is exactly one node that has no ancestors. ________

   (c) In a binary search tree with n keys, the *worst case* time to find a key is $O(\log n)$. ________

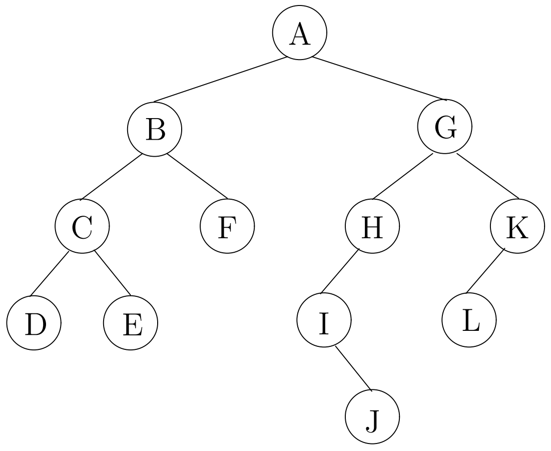
   (d) In a binary search tree with n keys, the *best case* time to find a key is $O(\log n)$. ________

   (e) If n is an arbitrary node in a tree T then then the sum of the height and depth of n equals the height of T. ________

   (f) In any red-black tree, the number of black nodes is always less than the number of red nodes. ________

   (g) In any red-black tree, the number of black nodes is always less than twice the number of red nodes. ________

3. (10 pts.) For each node in the following binary tree, determine the heights of its left and right subtrees, and whether the node is AVL balanced. Indicate the answers for each node by filling in the blanks in the appropriate row of the table that follows.



node	left subtree height	right subtree height	AVL balanced? (yes/no)
A			
B			
C			
D			
E			
F			
G			
H			
I			
J			
K			
L			

4. (10 pts.) Provide Java code[1] for a *nonrecursive* implementation of the *removeMin* operation on binary search trees. (Recall that *removeMin*(n) removes the smallest key in the subtree rooted at node n.) Your implementation should use a fixed (small constant) amount of working space (apart from the space used to store the tree), independent of the tree size.

   You may assume the availability of one or more pieces of code from the textbook; indicate the ones, if any, that you use by noting their figure and page numbers.

[1]or pseudocode *with a commensurate level of detail.*

5. (10 pts.) Suppose we modify the Nearest Common Ancestors (NCA) algorithm described in the textbook[2] to use a preorder traversal instead of a postorder traversal. Is the resulting algorithm correct? If so, explain why. If not, provide an input on which the modified algorithm produces incorrect results, explaining the results.

[2]Mark Allen Weiss, *Data Structures and Problem Solving Using Java*, 3rd edition (Addison-Wesley, 2006), Figure 24.11, p. 842.

6. (15 pts.) What is the *minimum* number of insert operations required to produce a *bottom-up red-black tree* with *at least* four black nodes? Reminder: Use the exact definitions and operations from the textbook.

   Justify your answer by providing the sequence of operations and the state of the red-black tree after each operation in the sequence. Clearly indicate the rotations that are performed following each insertion. Explain why fewer operations will not suffice to generate four black nodes.

   Use circles for red nodes and boxes for black nodes, as in class.

[additional space for answering the question on the previous page]

7. (10 pts.) ⋆ What is the *maximum* number of insert operations that may be applied to a *bottom-up red-black tree* (with no other operations permitted) to yield a bottom-up red-black tree with *at most* four black nodes? Reminder: Use the exact definitions and operations from the textbook.

   As in the previous question, justify your answer by providing the sequence of operations and the state of the red-black tree after each operation in the sequence. Clearly indicate the rotations that are performed following each insertion. Explain why any longer sequence of insert operations produces more than four black nodes.

   Use circles for red nodes and boxes for black nodes, as in class.

[additional space for answering the question on the previous page]