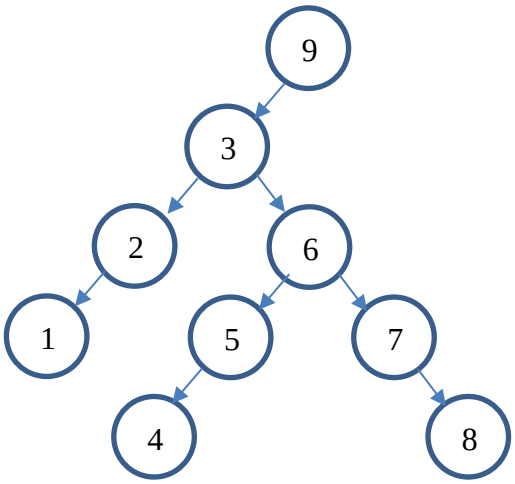
EECS 233 Homework 6 (Latest update: specified use of a binary search tree Problem #2.)

General requirements:

- Due at 11:00 PM on the posted due date.
- Include your name and network ID as a comment at the top of all of your programs.
- Create a typed document (.docx or .pdf) for the report with your name and network ID at the top.
- Upload your document and all .java files as a .zip file to Canvas. Do not use other formats such as .rar.
- All work should be your own, as explained in the Academic Integrity policy from the syllabus. File sharing is prohibited.

WARNING: There are various implementations of binary trees available on the Internet. However, students are required to use only the techniques that have been presented in class and in the following instructions. Additionally, students will be tested on their understanding of these particular techniques. Developing the programs on your own will maximize your understanding.

1. In your report, write the node values for the given tree in the following orders, either horizontally or vertically. Note that problem #3 will allow you to check your answers, but you should be able to do it without the help of a program.
   a. Pre-order
   b. In-order
   c. Post-order



2. In your report, draw the state of a binary search tree after each of the following (in order a – l):
   a. Insert 6
   b. Insert 1
   c. Insert 3
   d. Insert 2
   e. Insert 9
   f. Insert 7
   g. Insert 5
   h. Insert 4
   i. Insert 8
   j. Remove 4
   k. Remove 1
   l. Remove 6

3. Revise IntTreeBag2.java such that it implements a class for a binary search tree with the following methods:
   a. add() receives an integer value and adds it to the tree using the algorithm described in section 9.5 of the textbook and discussed in class. You are free to make trivial changes, such as with variable names, but do not deviate from the core algorithm.
   b. remove() receives an integer value and removes it from the tree using the algorithm described in section 9.5 of the textbook and discussed in class. You are free to make trivial changes, such as with variable names, but do not deviate from the core algorithm.
   c. preorderPrint() calls preorderPrint() for the root node.
   d. inorderPrint() calls inorderPrint() for the root node.
   e. postorderPrint() calls inorderPrint() for the root node.
   f. main() creates the tree produced in problem #2 above and prints it using the three print methods in (c), (d), and (e). Include the output in your report.

Rubric is forthcoming...