CS 3130 Design and Analysis of Algorithms Fall 2015

Programming Project #3 [80 points].

Due date: Wednesday, December 2.

A. Write a program implementing some operations on binary search trees. Requirements to your program:
   - The program should be menu-driven. The user will enter the code of operation. In the output, show that all functions work properly.
   - You can use the code of all tree functions from your textbook.
   - In the beginning of the program, create an array of integers from the following list: **30, 10, 45, 38, 20, 50, 25, 33, 8, 12** (in this particular order). Create a binary search tree with nodes containing these numbers as key values (here the function **TREE-INSERT** must be used).
- Display the results of *inorder*, *postorder* and *preorder* traversals of your binary tree.
- Write the function to find the height of the binary search tree and use it to determine the height of your tree.
- Show the result of **TREE-SEARCH** function calls for the keys **38** and **9**. Display search sequences in both cases.
- Delete the node with the key **10** (use the function **TREE-DELETE** or any other function that deletes the node promptly).
- Display the results of *inorder*, *postorder* and *preorder* traversals of your new binary search tree.

[45 points] Submit a 'script' file with your code and output as it is required in part A. No written analysis is needed.

= =

B. Write a program that builds *t* BSTs by inserting *N* keys into an initially empty tree, and then finds the tree height for *N*=100, 500 and 1000; and *t*=5, 10, 15. Find the average height of binary search trees for each pair of values of *t* and *N*. Decide what you will do with duplicates.

[25 points] Submit 'script' file for part B with your code and output that shows how your program works for ONE pair of *t* and *N*.

[10 points] Submit complete results of your work in the form of the table in the class or via e-mail. Include the description of the procedure of handling duplicates and theoretical efficiency of the function that is supposed to find the height of a binary search tree.
