CS 3130 Design and Analysis of Algorithms Spring 2020

Programming Project #3 [100 points].

Due date: Tuesday, April 21.

A. Write a program implementing some operations on binary search trees. Requirements to your program:

- You should implement the following operations on binary search trees: inserting an item; searching for a specific value; deleting a node; 3 types of traversals; finding the height of a tree; finding the smallest key in a given BST; finding the largest key in a given BST.
- In the output show that all functions work properly.
- You can use the code of tree functions from any source.
- 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.
- Display the results of *inorder*, *postorder* and *preorder* traversals of your binary tree.
- Determine the height of your tree.
- Find and display the smallest and the largest keys in your BST.
- Show the results of search for the keys **38** and **9**. Display search sequences in both cases.
- Delete the node with the key **10.**
- Display the results of *inorder*, *postorder* and *preorder* traversals of your new binary search tree.

[55 points] Submit a 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 random 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. Produce random numbers in the range 1 … 500. Decide what you will do with duplicates.

[35 points] Submit a 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 part B in the form of the table. 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.
