Data Structure: Assignment 3

Due: 2018/12/17 (Monday)

- Requirement: Implementation with C/C++ program
- Total: 100 points (**40 points** for questions 1-2 and **20 points** for question 3)

◆ Submission instructions:

[1]. Write a "**README file"** including the answers to problems 1-2, and a detailed note about the functionality of each of the above programs, and complete instructions on how to run them.

[2]. Make sure you include your name in each program and in the README file. **Each question has one C/C++ program file and one README file.** Make sure all your programs are fully commented, and compile and run correctly on the Linux-based machines.

[3]. Note that you **do not need to write a program for problem 3**. Instead, you should include your answer file containing the AVL tree you drew in your homework submission.

[4]. Submit your assignment to the portal system by the due date.

---

1. Implement a binary search tree ADT what include functions for performing the following operations:
   [1]. Insert a key
   [2]. Delete a key
   [3]. Find the minimum key in current binary tree
   [4]. Find the maximum key in current binary tree
   [5]. Print out current binary tree
   [6]. Perform a pre-order traversal of the tree
   [7]. Perform a post-order traversal of the tree
   [8]. Swap the left child and right child of every node of the binary search tree.
   What is the running time of your algorithm for each case?

2. Implement a heap tree structure to resolve the fractional knapsack problem. In this problem, the weights and values of n items will be given and then you need to put these items in a knapsack of capacity ***W*** to get the maximum total value in the knapsack.

   Input format:

   First line consists of two integers ***N*** and ***W***, denoting number of items and weight respectively.

Second line consists of 2*N spaced integers denoting Values and weight respectively.

Output format:

Print the maximum weight possible to put items in a knapsack, with decimal place of 2.

- **Example**

Input:

3 50

60 10 100 20 120 30

Output:

240.00

3. Show what the AVL tree looks like when the following numbers are inserted into initially empty AVL tree (please show the tree, the updated height information, and the updated balanced factor at the path from the insertion point to root node after each insertion).

   Insertion sequence: 43, 21, 88, 13, 12, 6, 25, 15