HO CHI MINH UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY
SOFTWARE ENGINEERING DEPARTMENT
ADVANCED PROGRAM IN COMPUTER SCIENCE
COURSE: **DATA STRUCTURE**
LECTURER: Dr. ĐINH BÁ TIẾN

WEEK 05
AVL TREE – INSERTION

- TRƯƠNG PHƯỚC LỘC
- HỒ TUẤN THANH

HCMC, 2016

TABLE OF CONTENTS

1 Paper assignments

As its title, you should complete these exercises on papers.
Note that you have to write your id and your fullname at the left-top corners of papers and the page numbers at the right-bottom corners of papers.

AVL trees named after 2 Russian mathematicians, Georgii Adelson-Velsky (1922 - 2014) and Evgenii Mikhailovich Landis (1921-1997).
AVL trees are height-balanced binary search trees. For every node, heights of left and right subtrees can differ by NO MORE THAN 1. Current height should be stored in each node. Balance factor of a node = height of left subtree – height of right subtree.
Insert operation may cause balance factor to become 2 or -2 for some node. Only nodes on the path from insertion node to root node have possibly changed in height. So after insertion, we go back up to the root node, update heights. If the a new balance factor is 2 or -2, adjust tree by rotation around the node.
Take a look at a reference here.
Link: http://courses.cs.washington.edu/courses/cse373/06sp/handouts/lecture12.pdf

1. Build an AVL tree with the following values: 15, 20, 24, 10, 13, 7, 30, 36, 25
2. Build an AVL tree with the following values: A, Z, B, Y, C, X, D, W, E, V, F

2 Coding assignments

2.1 Exercise 01 – AVL tree of integer numbers

1. Initialize an empty AVL tree
2. Insert x into the AVL tree, if it is not in that AVL tree yet
3. Search x in the AVL tree
4. Print out value, height, and balance factor of each node in an AVL tree in 3 ways:
   a. Pre-order (Node – Left – Right)
   b. In-order (Left – Node – Right)
   c. Post-order (Left – Right – Node)

2.2 Exercise 02 – Propose an data structure

Ms Thao, working at the APCS's academic affairs office wants to manage list of her students' records. For each student, we would like to mark whether he is a boy or she is a girl. Suggest a data structure that supports the following operations in O(logn) time in the worst case when the operation is executed (n is the number of students).

1. Insert(k,s) – Insert a new student s with id is k to data structure
2. Update(k) – Update info of student with id is k
3. FindDiff(k) – Find the difference between the number of girls and the number of boys among all the students with id smaller than k