LEBANESE AMERICAN UNIVERSITY

School of Arts and Science

Department of Computer Science and Mathematics

CSC 310: Algorithms and Data Structures

Lab 3

# Problem 1 –

Implement the class **AVLNode** which represents an AVL tree node having an *integer value, integer height and references to the left child and right child*, as well as a constructor that takes an integer as argument. Using AVLNode, implement the class AVL representing an AVL tree.

In the AVL class, implement the insert method, which takes as input an integer value and adds it to the tree maintaining the AVL property of the tree by performing the correct rotations (for every node, require heights of left & right children to differ by at most +1 or -1).

Given a sequence of integers, insert them into the constructed AVL tree maintaining the AVL property of the tree by using rotations (***for every node, require heights of left & right children to differ by at most +1 or -1***), then sort the array by traversing the constructed AVL tree using **Inorder Traversal**

The first line of input contains one integer N (the size of the array) and the second line of input contains *N* integers where the *ith* integer is the element to be inserted to the tree.

|  |  |
| --- | --- |
| **Sample Input** | **Sample Output** |
| **7** |  |
| 25 13 10 30 8 27 37 | 8 10 13 25 27 30 37 |
|  |  |

# Problem 2 –

Given a sequence of integers, insert them into a BST and check whether the tree is AVL or not. Print “**isAvl**” if the tree is AVL, otherwise print “**notAvl**”.

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| **7** |  |
| 25 13 10 30 8 27 37 | notAvl |

# Problem 3 –

Write a program that reads an array of integers and inserts them into a binary search tree. Then you are required to compare it with the AVL by running and measuring the performance of these programs multiple times for large input sizes.

Submit your work on the judge and put command for your results.