Course 12-746
Python Prototyping for Infrastructure Systems

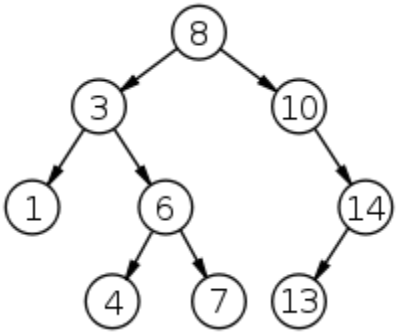
Fall 2016

Assignment 3 (Total: 8 points)

Due: Tuesday, Sept. 27th, by blackboard by the end of day. For each day the assignment is late, a penalty of 10% of your total grade will be deducted.

Task 1 (3.5 point) *More on Abstract Data Type (ADT): Tree*

A *tree* is a widely used abstract data type in computer science. It contains a set of linked nodes, with a value assigned to each node. And each node may have many "children" nodes, while only have one "parent" node. For more about *tree*, you can reference to the Wikipedia page of *tree*.
In this task, you will implement a special kind of *tree* called *binary search tree (BST)*, in which each node also has its own **value**, but can only has at most two children (we call them *left-child* and *right-child*), and the value of each node must be greater than all values of nodes in the left sub-tree and smaller than all values of nodes in the right sub-tree. The figure below shows one example of such tree. The root note has value as 8, which is larger than all subnotes on the left, and smaller than all notes on the right. The node with value 3 also has the same behavior.



Also, you may need to read the Wikipedia page for *binary tree* and *binary search tree* before you do this task.

Please use the task1.py file for this assignment.

a. Complete __init__ method in the class *binaryTreeNode* in file *task1.py* to represent the nodes in your binary search tree. Remember for each node in a binary (search) tree, it should have 3 attributes: *value, leftchild,* and *rightchild*. (0.5 pt)

b. The insertNode method is partially written. It can correctly handle the situation adding a node that has a value that is smaller than the current node. Complete the

insertNode method to handle the situation when inserting a value that is larger than current node. (1 pt)

c. The outputPreOrder method provides an example of Tree Traversal. If you do an In-Order travel, it will actually print all number in order from smallest to the largest. Please write a new method outputInOrder, and implement in-order traversal.
(1 pt)

d. Application: use the class *myBST* you just created to correctly store a list of integers, and output them in pre-order and in-order. (1 pt)
(For the whole task, you do not need to revise any existed code in the file *task1.py*, just complete the missing lines!)

Task 2 (2.5 point) *Implement myDate class in python*

Please use the myDate.py file for this assignment.

In this task, you will implement your own myDate class that behaves similar to the built-in Python Date class.

Here are the steps you need to go through in this task:

a. Complete the __init__ method to assign initial values to year, month and date. Also check whether the values are valid or not. For example, there should not be more than 30 days in April. (1 pt)

b. Complete the isLeapYr, DaysinMon, __eq__ , __cmp__, __sub__ and __repr__ method. (1.5 pt)

Complete those functions in myDate.py and verify them by examples

Task 3 (2 point) *Social Network Application*

In this question, you will build a small social network prototype. In this network, users can make friends with other users, check the complete friend list of their friends and view the list of mutual friends with an unfriended user. Please use the two Python modules provided in Assignment3_Task3.zip for this questions.

a. Complete the function *requestProfile* in file *User.py*
b. Complete the functions *friend* and *unfriend* in file *Network.py*
c. Check the correctness of the functions, you may try other cases too.