Computer Science

COMPSCI 105

Assignment TWO

Due: **11:59 pm Friday 3 June 2016**
Worth: **4% of the final mark**

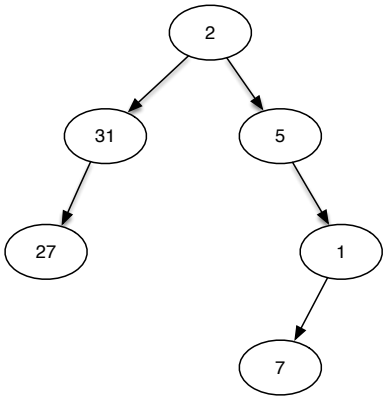
Introduction - Trees

In this assignment, you will implement the class RefBinaryTree as described in the Lecture 28. You will also implement a number of functions that operate on binary trees. ALL THE TREES DESCRIBED IN THIS ASSIGNMENT ARE BINARY TREES.

What you are to do

Stage 1: Implement the class RefBinaryTree as described in the Lecture 28. (10 marks)

Your implementation must provide, at a minimum, the following: __~~ini~~__(self, data), insert__left(self, data), insert_right(self, data), set_value(self, val), get_value(self), get_right_subtree(), get_left_subtree(). You are free to provide additionals methods as well. ~~You~~ must also provide a test function, *test_tree(tree)*, which returns the following tree:



Stage 2: Implement some conversion functions (15 marks)

tree_to_list(tree) returns a list of the tree node values. the list is in level order (top to bottom, left to right), if a child is missing then in its location, "None" is inserted. For example, the list produced from the tree shown above would be: [2, 31, 5, 27, None, None, 1, None, None, None, None, None, None, 7, None, end].

list_to_tree(list) returns a tree with the list values laid out in level order, values of "~~empty~~" mean that there is no value at that spot. *list_to_tree(list)* and *tree_to_list(tree)* are inverses. If I set ***a*** to a binary tree, and execute: *b = tree_to_list(a)* and then *c = list_to_tree(b)* then ***c*** would be a tree that looked exactly like ***a***.

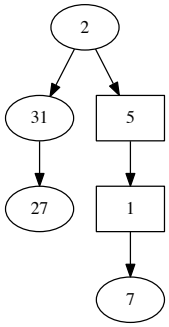
Stage 3: Print out trees using the "*dot*" language (15 marks)

print_tree(tree) prints out a *dot* presentation of *tree*. *dot* is a language for printing out graphs/trees. It is a very rich and powerful language, however, we are just going to use a very small subset of its features. *dot* files (e.g., "afile.dot") can be viewed using a program called *graphviz*. If *graphviz* is not on your computer then it can be downloaded from http://graphviz.org, you can also get documentation about the dot language there. The root and left nodes all have the default ellipse shape, the right nodes have a "box" shape.

The following is a *dot* language description of the tree above:

```
digraph tree2 {
    2 -> 31 ;
    2 -> 5;
    5[shape=box];
    31 -> 27;
    5 -> 1;
    1[shape=box];
    1 -> 7; }
```

The *graphviz* display of the above file is the following:



Submission

All code must be put into one file: *trees.py*

You may electronically submit your assignment through the Web Dropbox (https://adb.auckland.ac.nz/) at any time from the first submission date up until the final date. You can make more than one submission. However, every submission that you make replaces your previous submission. Only your very latest submission will be marked.

No marks will be awarded if your program does not compile and run.

DO NOT SUBMIT SOMEONE ELSE'S WORK:

- The work done on this assignment must be your own work. Think carefully about any problems you come across, and try to solve them yourself before you ask anyone for help.
- Under no circumstances should you take or pay for an electronic copy of someone else's work. This will be penalized heavily.
- Under no circumstances should you give a copy of your work to someone else
- The Computer Science department uses copy detection tools on the files you submit. If you copy from someone else, or allow someone else to copy from you, this copying will be detected and disciplinary action will be taken.
- To ensure you are not identified as cheating you should follow these points:
    - Always do individual assignments by yourself.
    - Never give another person your code.
    - Never put your code in a public place (e.g., forum, your web site).
    - Never leave your computer unattended. You are responsible for the security of your account.
    - Ensure you always remove your USB flash drive from the computer before you log off.