## COMP2402B (Fall 2015)

Carleton UNIVERSITY

## Assignment 5 of 5

**Due: December 7th, 2015**

# Caution

***You will be uploading your solutions to cuLearn but this assignment does have several written components.***

***Only the latest submission received before the deadline will be used to determine your mark on this assignment.***

*You must adhere to the following rules (as your submission may be subjected to automatic marking system):*

- ***Download** the compressed file **"COMP2402-F15-A5-(Start Skeleton).zip"** from cuLearn.*
- ***Retain the directory structure** (i.e., if you find a file in subfolder "comp2402a5", you must not remove it).*
- ***Retain the package directives** (i.e., if you see "package comp2402a5;" in a file, you must not remove it).*
- ***Do not rename any of the methods already present** in the files provided.*
- ***Do not change the visibility of any of the methods provided** (e.g., do not change private to public).*
- ***Do not change the main method of any class**; on occasion the main method provided will use command line arguments and/or open input and output files – you must not change this behaviour).*

*Please also note that **your code may be marked for efficiency as well as correctness** – this is accomplished by placing a hard limit on the amount of time your code wil be permitted for execution. If you select/apply your data structures correctly, your code will easily execute within the time limit, but **if your choice or usage of a data structure is incorrect**, your code may be **judged to be too slow** and **it may receive a grade of zero**.*

*You are expected to **demonstrate good programming practices at all times** (e.g., choosing appropriate variable names, provide comments in your code, etc.) and **your code may be penalized if it is poorly written**.*

### Written and Implementation Questions

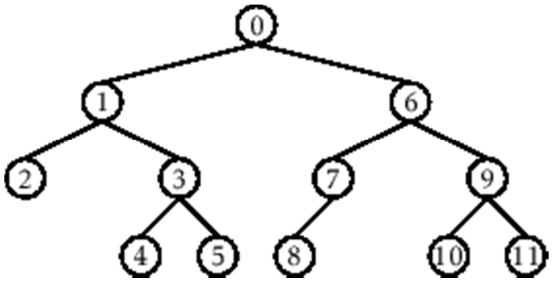
1. A pre-order traversal of a binary tree is a traversal that visits each node u before any of its subtrees. An in-order traversal visits u after visiting all the nodes in the left subtree of u but before visiting any of the nodes in the right subtree of u. A post-order traversal visits u only after visiting all other nodes in both subtrees of u.

   The pre/in/post-order numbering of a tree labels the nodes of a tree with the integers 0, ..., n-1 in the order that they are encountered by a pre/in/post-order traversal. See the following page for examples of pre/in/post-order numberings.

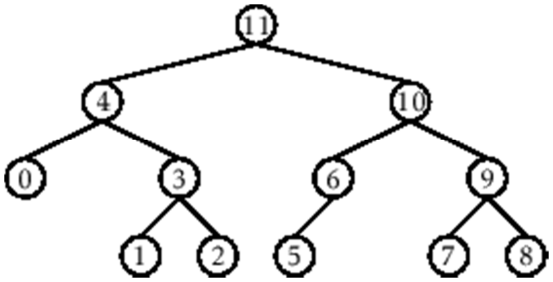
   For this question, you will use a BinaryTree class (as provided in the skeleton) whose nodes have fields for storing data (an Integer) as well as pre-order, post-order, and in-order numbers. You will write recursive methods `preOrderNumbers()`, `postOrderNumbers()`, and `inOrderNumbers()` that assign these numbers correctly.

   Revisit this task and write non-recursive methods `preOrderNumbersNonRecursive()`, `postOrderNumbersNonRecursive()`, and `inOrderNumbersNonRecursive()` that also assign these numbers, without using recursion in the same class.

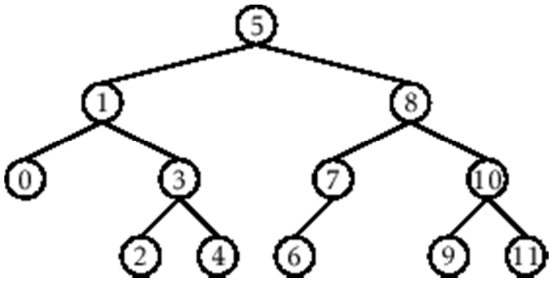
   All of these methods should each run in linear (i.e., O(n)) time.




Pre-Order Numbering




Post-Order Numbering




In-Order Numbering

2. Suppose we are given a binary tree with pre-order, post-order, and in-order numbers assigned to the nodes. Show (on paper) how these numbers can be used to answer each of the following questions in constant time:

    a. Given a node u, determine the size of the subtree rooted at u.

    b. Given a node u, determine the depth of u.

    c. Given two nodes u and w, determine if u is an ancestor of w.

3. A d-ary tree is a generalization of a binary tree in which each internal node has at most d children. Using Eytzinger's method it is also possible to represent complete d-ary trees using arrays. Work out the equations that, given an index u, determine the index of u's parent and each of u's children in this representation.

4. Using what you derived for Question 3, design and implement a DaryHeap class, the d-ary generalization of a BinaryHeap, using the skeleton code provided. Analyze the running times of the two main operations on a DaryHeap (add, remove and any secondary helper methods used for these) and test the performance of your DaryHeap implementation against that of the BinaryHeap implementation given here.

   For the comparison, you should compare instantiations of the BinaryHeap class and the DaryHeap class using several (at least 4) different values of d (that include d=2,3,5,10). You should compute the average time it takes to generate a large heap (262144 elements, averaged 20 times) and the average time (after creating a large heap) to perform a large number of removes and adds that roughly maintain the same size of heap. For the second comparison, after adding 262144 elements, measure the time to perform 131072 adds and 131072 removes, in different order combinations so that the size of the heap is roughly 262144, averaged over 20 trials. Include your code for generating this data in the main method of your DaryHeap class. Also include in your main method some tests (with appropriate comments and output) that check whether your DaryHeap is a correct implementation of the PriorityQueue interface.