American University of Armenia, CSE
CS121 Data Structures A, C
Fall 2021

Homework Assignment 5

Due Date: Saturday, November 13 by 23:59 electronically on Moodle

Please solve the programming tasks either in Java or C++, following good coding practices (details are on Moodle).

1. **(10 points)** Extend the `LinkedBinaryTree` class with a `grandchildren(`p`)` method that returns an iterable collection containing the grandchildren of position p (if any). Test it in a program. What is the running time of your method? Briefly justify your answer.

2. **(30 points)** Write two variants (recursive and iterative) of a generic method that, given a tree, calculates the number of non-`null` elements in it. Test both variants in a program using a `LinkedBinaryTree` object of size 10. What are the running times of your methods? Briefly justify your answer.

3. **(10 points)** Write a method, that given a partially filled array of `Integer`s representing a binary tree, checks if the tree is proper. Test the method in a program. Note that `null` elements in the array indicate the absence of the corresponding positions in the tree.

4. **(30 points)** Write a class `ArrayBinaryTree` that extends the `AbstractBinaryTree` class using a dynamic array as the underlying data structure. Note that the `AbstractBinaryTree` class in turn extends the `AbstractTree` class and implements the `BinaryTree` interface. Your class should support all of the following functionality:

   (a) two constructors that create an empty tree: a no-arg constructor that sets the default initial capacity of the array, and another constructor that receives the initial array capacity as an argument;

   (b) two methods for determining the **height** and **depth** of a given position and a method for determining the **height** of the tree;

   (c) functionality for traversing the elements of the tree, i.e. an `iterator()` method;

   (d) functionality for traversing the positions of the tree in preorder, postorder, inorder and breadth-first order traversals, i.e. `preorder()`, `postorder()`, `inorder()`, `breadthfirst()` methods, all of which return an iterable collection of the positions of the tree;

   (e) functionality for traversing the positions of the tree, i.e. a `positions()` method implementing preorder traversal;

   (f) methods `addRoot(`e`)`, `addLeft(`p, e`)`, `addRight(`p, e`)`, `remove(`p`)` similar to the corresponding methods for the `LinkedBinaryTree` class.

   Think carefully where you should add each of these methods; you may need to modify any of the `AbstractTree`, `AbstractBinaryTree` and `ArrayBinaryTree` classes.

   Specify and justify the running time of each method in your implementation.

5. (5 points) Draw the unique binary tree T, given the following inorder and postorder traversals of the elements of T:

inorder:	v	i	g	o	b	x	c	q	z	f	y	p
postorder:	i	o	g	v	c	x	b	y	p	f	z	q

6. (15 points) Extend the `ArrayBinaryTree` class with an **iterative** `inorderAfter` method that, given a position p in the tree, returns the position q that follows p in an inorder traversal of the tree.