Community College of Philadelphia

CSCI211 – Data Structures and Algorithms
ASSIGNMENT 9

Overview

Using C++, Java, or Python, design a program that creates a self-balancing (AVL or Red-Black) BST, where each node contains an object with information about a certain tropical cyclone.

Provided with the homework assignment is a CSV file containing entries for tropical cyclones to impact the United States mainland. This is a **DIFFERENT** CSV file than was used in the previous assignment.

Your program must read in the data contained in this CSV file to set the fields/attributes of the objects stored in your BST's nodes.

See the following pages of these instructions for details about the necessary classes/objects and an outline of what will happen in your main program.

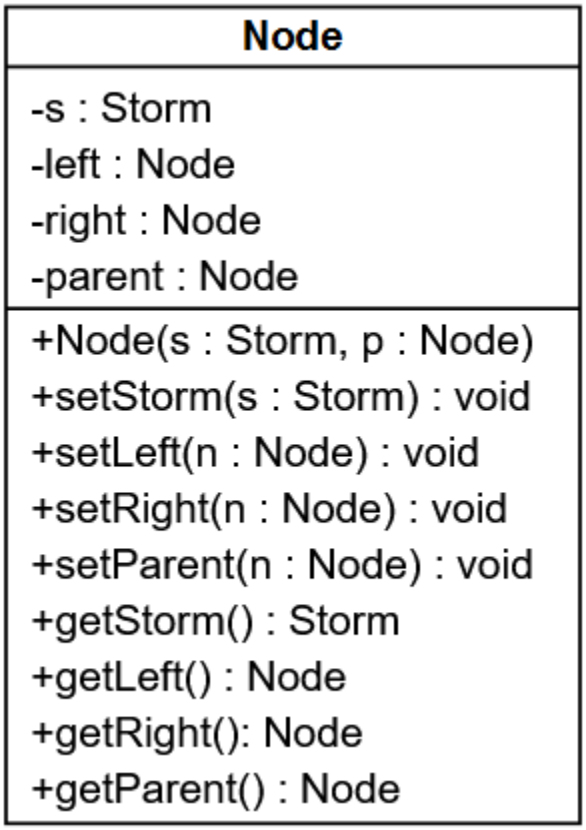
Object

Storm
-name : String -year : int -category : int -cost : float (or double)
+Storm(name : String, year : int, category : int, cost : float (or double)) +getName() : String +getCost() : float (or double)

The class can be implemented as either a struct (C++) or a class and must **only** contain:

- The private fields as identified in the class diagram.
- The public methods as identified in the class diagram.

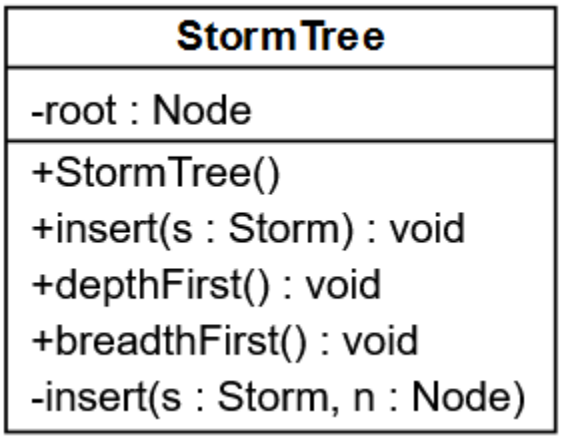
Node



The class can be implemented as either a struct (C++) or a class and must contain:

- The private fields as identified in the class diagram.
- The public methods as identified in the class diagram.
- You will also need:
    - A color field/setter/getter if using a Red-Black Tree
    - A height field/setter/getter if using an AVL Tree

Tree



The class can be implemented as a class and must **only** contain:

- The private field as identified in the class diagram.
- The methods as identified in the class diagram.
  - You'll need a public and private insert function. The private insert function will recursively insert the node at the correct position based on a Storm's **cost**.
  - The two traversal methods perform depth-first and breadth-first traversals.
  - You'll also need appropriate rotation functions, and any other auxiliary functions necessary for the type of tree you have selected.

Main Application

For each Storm object, you'll insert it into the BST that you have implemented. (Do not use tree-like data types built into the language). Insert them in the order in which they appear in the CSV file.

After all objects have been inserted, perform the following two traversals on your BST using the methods in the class:

- A depth-first traversal
- A breadth-first traversal

Print each Storm's name (not the cost) at the correct time in each traversal. For the traversals, you may use Queues/Stacks/LinkedLists that are built into the language.

Submit all related source code files in the Assignment 9 submission link.

Grading

See Assignment Rubric in Canvas.