**Fully functional Binary Tree class (100pts, +20 ec)**

**Project Description & Goal**

The goal of this project is to understand the basics of a binary tree and to practice recursion.

**Project Specifications**

* Create a class that represents a node in a binary tree.
* Create a class that represents a binary tree (aka, a container for binary tree nodes).
* Implement all required methods
* Test your implementations in Program.Main()

**Required Functionality**

* Binary tree implements
  + Insert
  + Search
  + Remove
  + PreOrderPrint
  + InOrderPrint
  + PostOrderPrint

**Project Learning Objectives**

* Understand a binary tree.
* Practice object oriented programming through the implementation of a binary tree.

**Project Demonstrated Competencies**

1. Binary tree node holds the correct information.
2. Binary tree holds the correct information
3. Binary tree has the correct functionality
4. Program.Main() tests binary tree’s functionality and proves that it works.
5. Optional: Make your binary tree a generic class, so it can be a tree of strings, ints, floats, etc.
6. Optional: Write an algorithm that balances your binary tree.

**Rubric**

|  |  |  |
| --- | --- | --- |
|  | **Description of perfect implementation** | **Score** |
| Competency #1 | The binary tree node class has information related to its purpose. Value, LeftChild, RightChild, etc. | \_\_\_  20 |
| Competency #2 | The binary tree class has information related to its purpose. Root, Height, Count, etc. | \_\_\_  20 |
| Competency #3 | The binary tree class implements all the required functionality. | \_\_\_  40 |
| Competency #4 | Program.Main() accurately tests the binary tree class functionality. | \_\_\_  20 |
| Competency #5 | Binary tree class is generic and can handle at least strings, chars, ints, and floats. | \_\_\_  +10 |
| Competency #6 | You can call tree.Balance() and your tree will balance itself. | \_\_\_  +10 |