CIS 211, S20 Lab 4 Name ____________________

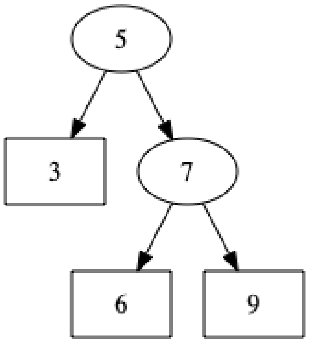
Overview

1. **Review of Dynamic Dispatch -** what does that mean and how to take advantage of it?
2. **Object-Oriented Recursion -** with an example of binary search tree (**BST**).

Exercise

We will be implementing three classes to represent binary search trees like Figure 1 shows. We will have abstract class `Node` and two concrete subclasses `Leaf` and `Internal`. We say node 3 is the parent of node 3 and node 7, and we say node 3 and node 7 are the children of node 5. Nodes in circle are of type `Internal`. They have children. Those in box are of type `Leaf` and they don't have children. Note that this is not the only way to implement BST.

Figure 1: A BST.



1 Make a `Python` file with whatever name you like. Create a class `Node`, whose constructor takes a parameter `value` of type `int`. Inside the constructor, create a class variable to save `value`. Make method `sum` in class `Node`, raising `NotImplementedError` with an error message. Method `sum` will sum the value of current node and all nodes below. For example, if we call `sum` on node 5 of Figure 1, it should give us 30.

2 Make a class `Leaf` inheriting `Node`, whose constructor takes a `value` as well. Make a class `Internal` inheriting `Node` as well, whose constructor takes `value: int`, `left: Node`, and `right: Node`. The `left` and `right` are left and right children of an `Internal` node.

3 Now implement `sum` method in both `Internal` and `Leaf`. Note that a child of an `Internal` node could be a `Leaf` node, like both children of node 7, or it could be another `Internal` node, like the right child of node 5. You can assume that both children of `Internal` are not `None`.

4 Add the following code to your file and run your program.

```
def main():
    l1 = Leaf(3)
    l2 = Leaf(6)
    l3 = Leaf(9)
    i = Internal(7, l2, l3)
    root = Internal(5, l1, i)
    print(root.sum())

if __name__ == '__main__':
    main()
```

5 At this point you should see the recursive nature of the `sum` method. Now implement `__str__` method in all three classes so that it gives us the representation of a tree or a subtree in the form of `<value, left, right>`. For example, if we add the following line to our `main` function,

```
    print(root)
```

we should get `<5, 3, <7, 6, 9>>`.