COSC 222: Data Structures
Lab 4 – Trees

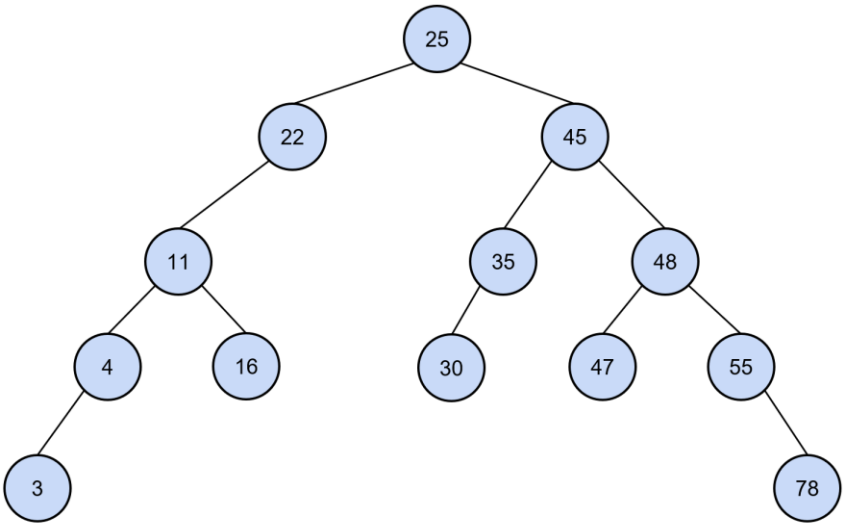
Question 1 (Tree Basics) [10 marks]
This question does not require coding. You will need to submit a PDF file with your answers. This can be handwritten and scanned or created using a paint application or word processor. Name your file **Lab4Question1.pdf**.

Part A [4 marks]:
Draw what a binary search tree would look like if the following values were added to an initially empty tree in this order. Partial marks will be awarded for wrong answers *if* you show your process step-by-step.

- **Tree 1:** 44, 20, 15, 25, 12
- **Tree 2:** 10, 25, 33, 41, 55, 45, 50
- **Tree 3:** 9, 12, 5, 29, 34, 7, 10, 28, 33, 27
- **Tree 4:** 12, 45, 15, 96, 34, 33, 35, 92, 48, 11

Part B [6 marks]:
Draw what will happen to this binary tree after deleting the specified node (Please use the steps discussed in the class lecture)

Original Tree:



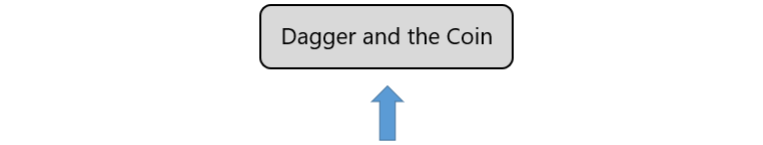
- From original tree - Delete Node 78 (1 mark)
- From original tree - Delete Node 4 (1 mark)
- From original tree - Delete Node 45 (2 marks)
- From original tree - Delete Node 25 (2 marks)

Question 2 [10 marks + (*2 marks BONUS*)]

Use the starter java files provided: **TreeNode.java**, **BinaryTree.java**, and **BinaryTreeTest.java**.

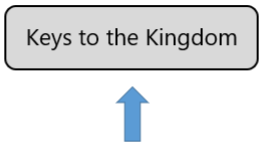
In this question, you will practice building a Binary Tree to solidify your understanding of the tree data structure. Begin with the **BinaryTree.java** file. In this file are several completed methods (3 constructors, `inOrderTraversal()`, and `inOrderRecursive()`), and several incomplete methods. You need to complete the following methods inside this file:

First, check the `BinaryTreeTest.java` file. If we call `BinaryTree ll = new BinaryTree("Dagger and the Coin")`, it creates a new tree called `ll` with a root that contains `"Dagger and the Coin"`.



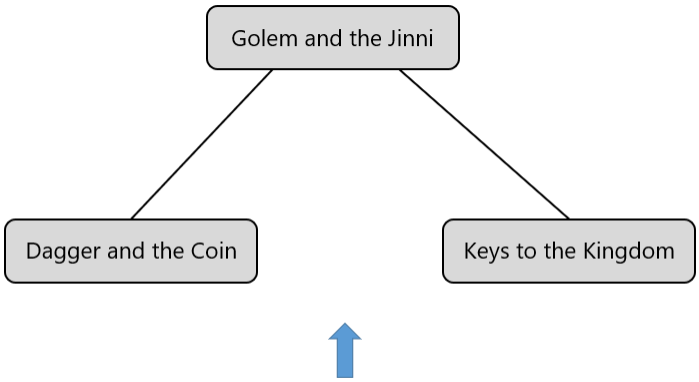
`BinaryTree ll = new BinaryTree("Dagger and the Coin")`

If we call `BinaryTree lr = new BinaryTree("Keys to the Kingdom")`, it creates a new tree called `lr` with a root that contains "Keys to the Kingdom".



`BinaryTree lr = new BinaryTree("Keys to the Kingdom")`

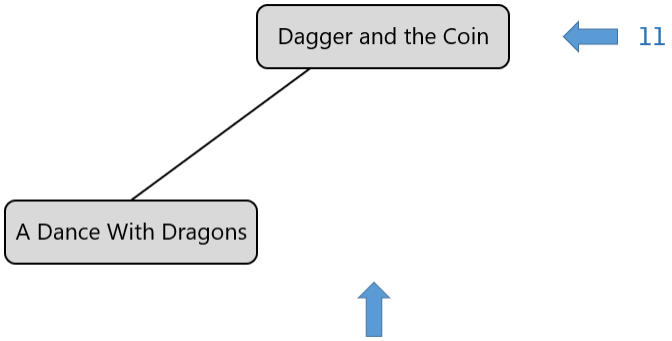
If we call `BinaryTree l = new BinaryTree("Golem and the Jinni", ll, lr)`, it creates a binary tree with a root (value `"Golem and the Jinni"`) and assigns `ll` as the left subtree and `lr` as the right subtree of the root.




```
BinaryTree l = new BinaryTree("Golem and the Jinni", ll, lr)
```

Part A [2 marks]:

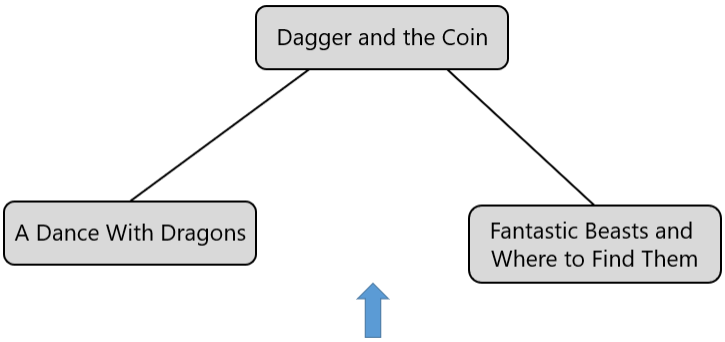
`attachToLeft()`: The first thing you'll want to do is create a way to add data to a tree. This method should add a new node with the given data to the left branch of the tree. If the left branch of the tree is already occupied, the method should show a message to inform users about this. Assume that we have a tree called `ll`, which only has a root that contains `"Dagger and the Coin"`. If we call `ll.attachToLeft("A Dance With Dragons")`, it will add a new node containing `"A Dance With Dragons"` and set the node as the left child of the root.




```
ll.attachToLeft("A Dance With Dragons")
```

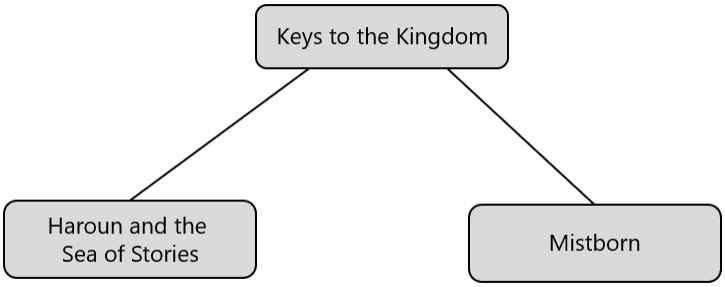
Part B [2 marks]:

`attachToRight()`: This method works exactly the same as the previous, except that it attaches the new node to the right branch, rather than the left. Again, show a message if the right node is already occupied. For the previous `ll` tree, if we call `ll.attachToRight("Fantastic Beasts and Where to Find Them")`, it will add a new node containing "`Fantastic Beasts and Where to Find Them`" and set the node as the right child of the root.

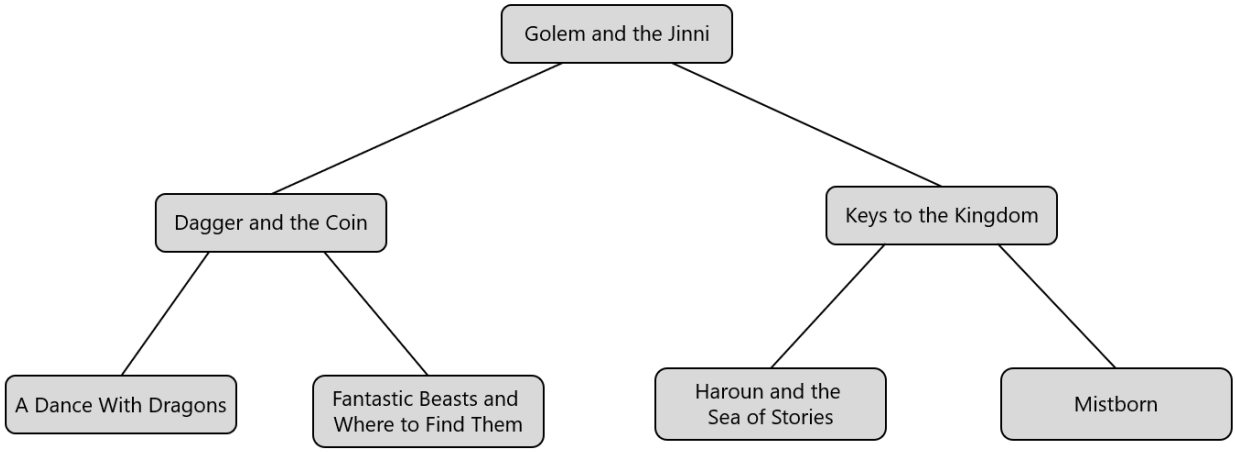


Note: We can create another tree by using executing the following code:

```
BinaryTree lr = new BinaryTree("Keys to the Kingdom");
lr.attachToLeft("Haroun and the Sea of Stories");
lr.attachToRight("Mistborn");
```



Now, we have two trees: `ll` and `lr`. We would like to add these two trees under another tree with a new root (value “Golem and the Jinni”). We can do this by calling `BinaryTree l = new BinaryTree(“Golem and the Jinni”, ll, lr)`. Here, a binary tree is created with a root value “Golem and the Jinni” and assigned `ll` as the left subtree (i.e., the left child pointer of the root holds the reference of `ll`) and `lr` as the right subtree of the root (i.e., the right child pointer of the root contains the reference of `lr`).




```
BinaryTree l = new BinaryTree("Golem and the Jinni", ll, lr)
```

Part C [2 marks]:

`attachToLeftSubtree()`: this method adds a sub-tree to the left branch of the tree. Show a message if the left branch is already occupied.

Part D [2 marks]:

`attachToRightSubtree()`: This method works the same as the `attachToLeftSubtree()` method, except that you will be attaching the new subtree to the right branch, rather than the left.

Part E [2 marks]:

`height()`: This method returns the height of the tree using recursion. (Hints: check lecture 7 Tree Part 2 slides 19 – 23 for details)

Part F [2 marks (BONUS)]:

`size()`: This method returns the number of nodes in the tree using recursion. (Hints: check for conditions where (i) a node doesn't have any child, (ii) has a left child, or (iii) has a right child or, (iv) has both left and right child)

Sample Output:

```
-----Test Tree 1-----
Inorder traversal:
A Dance With Dragons
Dagger and the Coin
Fantastic Beasts and Where to Find Them
Golem and the Jinni
Haroun and the Sea of Stories
Keys to the Kingdom
Mistborn
```

```
Name of the Wind
On Stranger Tides
Rage of Dragons
Wheel of Time

Size of the tree: 11
Height of the tree: 3

-----Test Tree 2-----
Inorder traversal:
Lord of The Rings
The Chronicles of Narnia
Voyage to Arcturus
Wardstone Chronicles

Size of the tree: 4
Height of the tree: 2

-----Test Tree 3-----
Inorder traversal:
A
D
G
M
P
S
W

Size of the tree: 7
Height of the tree: 2
```

Submission Instructions:

- Create a folder called "Lab4_<*student_ID* >" where <*student_ID* > is your student number/ID (e.g. **Lab4_12345678**). Only include the mentioned java files in the folder. **DO NOT** include any other files (e.g., .class, .java~ or any other files)
    - For question 1, Include **Lab4Question1.pdf** file.
    - For question 2, Include your **BinaryTree.java** file.
- Make a **zip file** of the folder and upload it to Canvas.
- To be eligible to earn full marks, your Java programs **must compile and run** upon download, without requiring any modifications.
- These assignments are your chance to learn the material for the exams. **Code your assignments independently.**