- It is assumed that you have reviewed chapter 6 & 7 of the textbook. You may want to refer to the text and your lecture notes during the lab as you solve the problems.
- When approaching the problems, think before you code. Doing so is good practice and can help you lay out possible solutions.
- Think of any possible test cases that can potentially cause your solution to fail!
- You must stay for the duration of the lab. If you finish early, you may help other students. If you don't finish by the end of the lab, we recommend you complete it on your own time. Ideally you should not spend more time than suggested for each problem.
- Your TAs are available to answer questions in lab, during office hours, and on Piazza.

Vitamins

---

1. Draw the following trees and answer the associated questions:

    a. Given the following traversal of a **binary tree**. Restore the tree:

    Preorder:
    ```
    7 4 3 8 9 1 6 5 2
    ```

    Inorder:
    ```
    3 8 4 7 1 6 9 5 2
    ```

    b. If you are given just the preorder, can this **binary tree** be unique? If not, give a counterexample (two different trees with this same preorder).

    Preorder:
    ```
    5 2 1 3 4 9 7 6 8
    ```

    c. Now, if you are given just the preorder of a **binary search tree**, will this tree be unique?

    Preorder:
    ```
    5 2 1 3 4 9 7 6 8
    ```

2. Thanos suggested the following implementation for checking whether a LinkedBinaryTree object is also a Binary Search Tree.

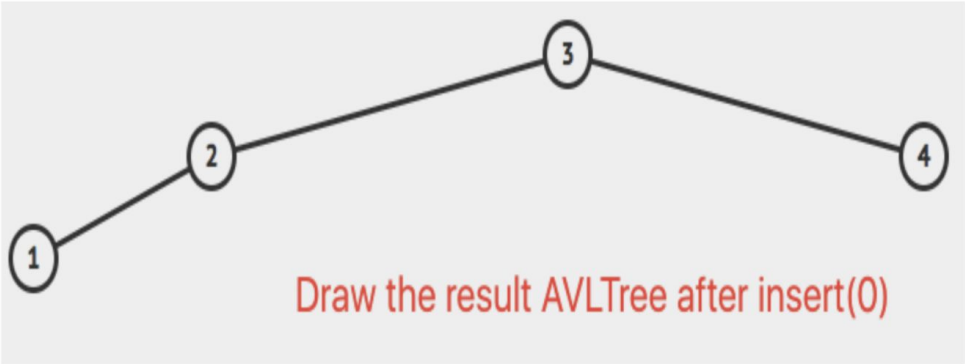
```
def is_BST(root):

    if (root.left is None and root.right is None):
        return True

    elif root.left and root.right:
        check_left = root.left.data < root.data
        check_right = root.right.data > root.data
        return check_left and check_right and is_BST(root.left)
and is_BST(root.right)

    elif root.left:
        check_left = root.left.data < root.data
        return check_left and is_BST(root.left)

    elif root.right:
        check_right = root.right.data > root.data
        return check_right and is_BST(root.right)
```

Is there a problem with this function? If so, draw a simple binary tree that will cause this function to return an incorrect result.

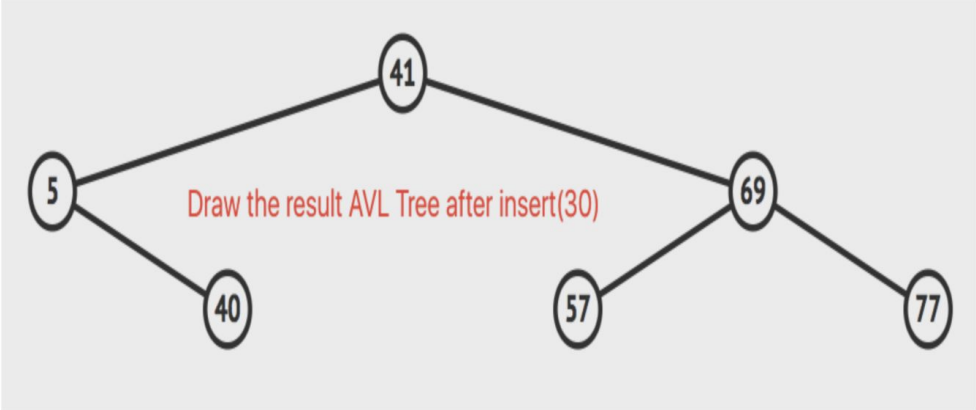
ex) either draw **a BST** that makes is_BST return **False**

or a **non BST** that makes is_BST return **True**.

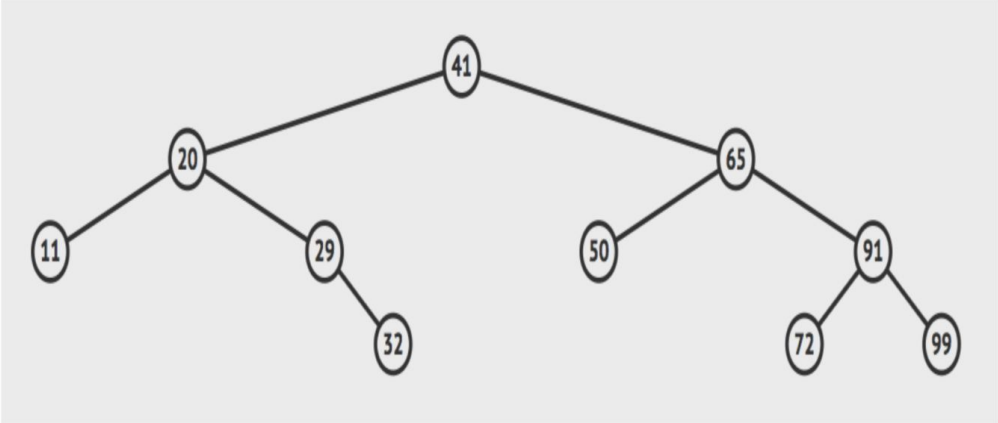
3. Draw the following



a). Suppose we have the AVLTree above. Draw the AVLTree after insert(0)



b). Suppose we have the AVLTree above. Draw the AVLTree after insert(30).



c). Suppose we have the AVLTree above. Draw the AVLTree after insert(73).

Coding

---

1. Write a *recursive* function that returns the sum of all even integers in a `LinkedBinaryTree`. Your function should take one parameter, LinkedBinaryTree . You may assume that the tree only contains integers.

```
def bt_even_sum(binTree):
    ''' Returns the sum of all even integers in the binary
    tree'''
```

2. Write a *recursive* function that determines whether or not a value exists in a `LinkedBinaryTree`. Your function should take two parameters, a *root* node and *val*, and return True if *val* exists or False if not.

```
def bt_contains(root, val):
    ''' Returns True if val exists in the binary tree and
    false if not'''
```

3. Let's fix the is_BST function that was incorrectly defined in the vitamins section of this lab. Given the root node of a LinkedBinaryTree, implement a function that will return True if the tree is a Binary Search Tree and False if not.

   The helper function should return a tuple triplet containing the min and max of the current subtree, and a bool value. (min, max, bool)

```
def is_BST(root):
     return is_BST_helper(root)[2]


def is_BST_helper(root):
''' Returns a tuple (min, max, bool)'''
```