NAME ____ **ICA-18**

NetID ____________________@email.arizona.edu

Problem 1. Tree terminology. In addition to the terms introduced in class, the following terms are defined as follows:

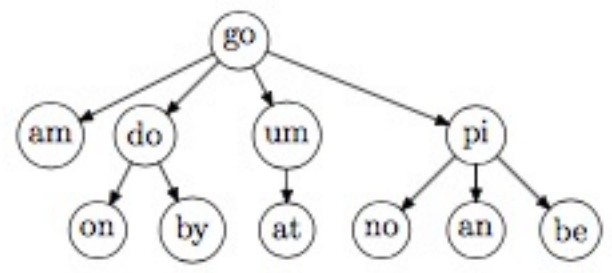
sibling – a set of nodes that all share the same parent

degree – the number of children a node has

level – the number of "links" from the root to a node

height – the maximum of the levels of the nodes in the tree.

Answer the questions below given the tree pictured.



What are the leaves of this tree?	______________________________
Which nodes have multiple children?	______________________________
Which nodes have multiple parents?	______________________________
What is the root of this tree?	______________________________
Which nodes are siblings of 'um'?	______________________________
What is the level of 'at'?	______________________________
What is the height of this tree?	______________________________

Problem 2. As shown in lecture, the class definition of a BinaryTree is:

```
class BinaryTree:
    def __init__(self, value):
        self._value = value
        self._lchild = None
        self._rchild = None
```

Write a method `insert_right(self, value)` that inserts a BinaryTree node in the right child of a binary tree.

Hint: There are two cases for insertion.

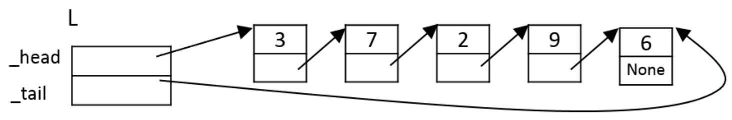
First, the node may have no existing right child. In that case, simply create a new BinaryTree and add it to the tree as the right child.

In the second case, there is an existing right child. Just as with adding a node to a linked list, you must be careful to get the order of the assignments correct so that you don't overwrite the reference in the existing right child before using it.

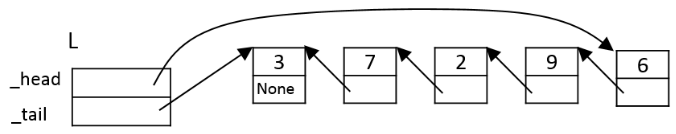
Problem 3. What is a BST? Draw an example. How does it differ from a binary tree?

Devise an algorithm, just in words, that explains how to start at the root of a binary search tree and locate the largest value in the tree.

Problem 4.
Consider a Python linked list class with both head and tail references. Write a method **`reverse()`** for this class to reverse a list. For example, given the following LinkedList object L:

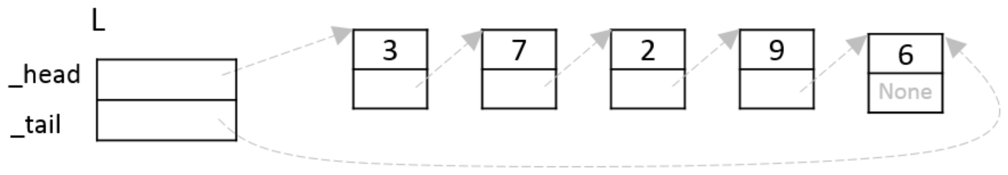


executing **`L.reverse()`** should result in the following:



To do this, begin by working out the steps of your algorithm on the following picture, then write the code. Finally, step through the code on this picture to make sure that it works as expected.

Note: If necessary you can use some additional variables in your code, but the number of such variables should not depend on the length of the list being reversed.



Problem 5 [***quicksort***].
You have to sort a given a list of integers L. Suppose you could create two lists:

- a list of all the values in L that are smaller than or equal to L[0], and
- another list of all the values in L that are larger than L[0].

How could you use this to write a recursive function to sort L?

Specific points to consider:

1. What are the values used by the computation?
2. When does repetition stop?
3. What is the work done in each round of repetition?
4. How do you set things up for the next round of repetition?