Quiz #9: Binary Search Trees

Read over the declaration of class `BinarySearchTree`, and the header and docstring for method `num_in_range`:

```python
class BinarySearchTree:
    """A Binary Search Tree.
    This class represents a binary tree satisfying the Binary Search Tree
    property: for every node, its value is >= all items stored in its left
    subtree, and <= all items stored in its right subtree.
    """

    --- Private Attributes ---
    @type _root: object
    The item stored at the root of the tree, or None if the tree is empty.
    @type _left: BinarySearchTree | None
    The left subtree, or None if the tree is empty
    @type _right: BinarySearchTree | None
    The right subtree, or None if the tree is empty

    --- Representation Invariants ---
    - If _root is None, then so are _left and _right. This represents an empty BST.
    - If _root is not None, then _left and _right are BSTs.
    - All items in _left are <= _root, and all items in _right are >= _root.
    """

def num_in_range(self, start, end):
    """Return the number of items between <start> and <end>, inclusive.
    Use the BST property to ensure you don’t make unnecessary
    recursive calls.
    ""
    @type self: BinarySearchTree
    @type start: int
    @type end: int
    @rtype: int
    """

Below is a picture of a binary search tree, with several levels. We'll refer to this tree as the variable bst.

![Binary Search Tree Diagram]
1. What is the output of \texttt{bst.num\_in\_range(5, 50)}?

2. What are the outputs of \texttt{bst._left.num\_in\_range(5, 50)} and \texttt{bst._right.num\_in\_range(5, 50)}?

3. Give an example of numbers \(x\) and \(y\) such that the call \texttt{bst.num\_in\_range(x, y)} would only require \textit{one} subtree to be checked.

4. Implement \texttt{num\_in\_range} using only the attributes \_root, \_left, \_right, and the method \texttt{is\_empty}.

```python
def num\_in\_range(self, start, end):
```