Questions about binary trees refer to the definitions on the last page for **BinaryTree**, **BinarySearchTreeInterface**, **BinarySearchTree**, and **Node**.

1. [4] Briefly explain why the **root** data member in the **BinaryTree** class is declared to be **protected**?

2. [4] Briefly explain why the **Node** class is declared to be **static**?

3. [10] Write a **recursive** **BinaryTree** method named **count_leaves** that returns the number of leaves in the tree.
4. [4] Draw the binary search tree that results from inserting the integers 57, 85, 35, 9, 47, 20, 26, 99, 93, 10 starting with 57 and ending with 10.

5. [4] What is the preorder traversal of your tree from question 4?

6. [4] What is the postorder traversal of your tree from question 4?

7. [4] What is the breadth first traversal of your tree from question 4?


9. [3] The _______________ data structure is needed for an iterative (non-recursive) function that does a preorder traversal of a binary tree. (I want the best and most precise answer. Don't write binary tree)
10. [3] The _______________ data structure is needed for the iterative (non-recursive) function that does a breadth first traversal of a binary tree. (I want the best and most precise answer. Don't write binary tree)

11. [10] Write an iterative (non-recursive) **BinarySearchTree** method named **min** that returns the smallest value in a binary search tree.

12. [4] Consider the binary search tree below where **root** refers to 70. On the line below write a single Java statement that will update the node that has 53 in it to contain a 58.

```
70
 / 
11 75
 /   
10 47
 /   / 
12 20 53
 /   /   
12 62
```

__________________________
13. [4] Using the binary search tree in question 12, redraw the tree after deleting 70.

14. [4] If we insert \( n \) items into a binary search tree and the resulting tree has a height of \( n \), what does that say about the items?

15. [4] The **minimum** number of leaves in a binary search tree of height \( h \) is ______________

16. [4] The **maximum** number of leaves in a binary search tree of height \( h \) is ______________

17. [4] The **minimum** height of a binary search tree with 79 nodes is ______________

18. [3] The **worst case** time efficiency for inserting into a binary search tree with \( n \) nodes and height \( h \) is:
   
   a) \( O(\log(n)) \)
   
   b) \( O(h) \)
   
   c) \( O(1) \) (constant time)
   
   d) none of the above
   
   Answer:_______________

19. [3] The worst case time efficiency for the recursive preorder function on a binary tree with \( n \) nodes and height \( h \) is:

   a) \( O(\log(n)) \)
   
   b) \( O(h) \)

   c) \( O(n) \)

   d) none of the above

   Answer:_______________
20. [10] Given the mystery function below what is the value of mystery(7263)

```c
int mystery(int x) {
    if (x < 10)
        return 0; Answer: __________
    else
        return 1 + mystery(x/10);
}
```

21. [5] Consider the function below that recursively computes the sum of 1 to n. State a precondition for the function sum. (Looking for the best answer, For example, “n must be an integer” is not a good answer).

```c
int sum(int n) {
    if (n == 0)
        return 0;
    else
        return n + sum(n-1);
}
```
public class BinaryTree<E> {
    protected Node<E> root;

    // BinaryTree methods

    protected static class Node<E> {
        E data;
        Node<E> left;  // left subtree
        Node<E> right; // right subtree

        Node(E data) {
            this.data = data;
            left = right = null;
        }
    }

    interface BinarySearchTreeInterface<E> {
        E find(E data);
        void insert(E data);
        void delete(E data);
    }

    public class BinarySearchTree<E extends Comparable<E>>
        extends BinaryTree<E>
        implements BinarySearchTreeInterface<E> {

        // BinarySearchTree methods
    }
}