Data Structures - Test-2

Please complete the exam and submit it as a plain text email with the subject "CISC 3130 Test-2" to nzhou@brooklyn.cuny.edu by 4PM on Monday, May 3.

The class `ListNode` and the class `TreeNode` referred to in the questions are defined below:

```
class ListNode<E> {
    E data;
    ListNode<E> next;

    public ListNode(E data){
        this.element = data;
        next = null;
    }

    public ListNode(E data, ListNode<E> next){
        this.element = data;
        this.next = next;
    }
}

class TreeNode<E> {
    E element;
    TreeNode<E> left;
    TreeNode<E> right;

    public TreeNode(E element){
        this.element = element;
    }
}
```

Question 1:

Consider the function `f1` defined below:

```
static <E> ListNode<E> f1(ListNode<E> head, int index, E elm){
    ListNode<E> pre, cur;
    int i = index;

    if (index < 0){
        throw new IndexOutOfBoundsException(""+index);
    }
    if (index == 0 || head == null){
        return new ListNode<E>(elm, head);
    }
    pre = cur = head;
    while (cur != null && i > 0){
        pre = cur;
        cur = cur.next;
        i--;
    }
    pre.next = new ListNode<>(elm, cur);
    return head;
}
```

Assume `head` references the head of the list `[1,2,9,8,10]`. What is the return value of `f1(head, 2, 3)`?

Question 2:

Consider the following function `f2`:

```
static <T extends Comparable<T>> BTreeNode<T> f2(BTreeNode<T> root, T val) {
    if (root == null)
        return new BTreeNode<>(val);
    if (val.compareTo(root.element) == 0)
        throw new IllegalArgumentException("No Duplicates Allowed");
    else if (val.compareTo(root.element) < 0)
        root.left = f2(root.left, val);
    else
        root.right = f2(root.right, val);
    return root;
}
```

Assume `root` references the root of the following tree:

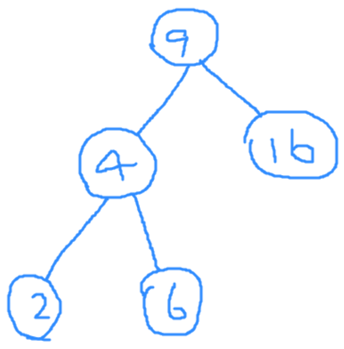

Figure 1: An example binary tree

Show the tree that `root` references after the following statements:

```
    root = f2(root, 3);
    root = f(root, 10);
```

You can describe the changes to the tree without drawing the resulting tree.

Question 3:

Write the following functions on `ListNode`:

- `public static <E> int len(ListNode<E> head)`: This function returns the number of nodes in the list whose first node is referenced by `head`.
- `public static <E> E last(ListNode<E> head)`: This function returns the last element in the list whose first node is referenced by `head`.
- `public static <E> E get(ListNode<E> head, int i)`: This function returns the element at index `i` in the list whose first node is referenced by `head`. The function throws `IndexOutOfBounndsException` if `i` is less than 0 or greater than `len(head)-1`.

Question 4:

Write the following functions on `TreeNode`:

- `public static <E> int countLeaves(TreeNode<E> head)`: This function returns the number of leaf nodes in the tree whose root is referenced by `head`. For example, for the tree in Fig. 1, the function returns 3.
- `public static <E> ArrayList<E> leftMostPath(TreeNode<E> head)`: This function returns a list of node values on the path from the root to the left-most leaf. For example, for the tree in Fig. 1, the function returns [9,4,2].

Question 5:

The following gives a function that tests if a tree is balanced:

```
public static <E> boolean balanced(TreeNode<E> root){
    if (root == null)
        return true;
    if(Math.abs((depth(root.left) - depth(root.right))) > 1){
        return false;
    }
    return balanced(root.left) && balanced(root.right);
}
```

The used algorithm is not efficient since it visits a node multiple times. Give an implementation of a linear-time algorithm for the function that visits each node exactly once.