Code for test2sample (answer 4b is revised)

// 3a. Write a print method in Java that takes a reference to Node<E> as
// its only parameter and prints the data in each node on a separate line.
// It should not change the linked list.
public static<E> void print(Node<E> head) {
    while (head != null) {
        System.out.println(head.getData( ));
        head = head.getLink( );
    }
}

// 3b. Draw a picture showing what happens in an addAfter method that adds
// the item (provided as a parameter) to a new node added after the node given
// as the parameter pre.
// 3c. Write the Java code for the addAfter method from part 3b. This is raw
// linked list not in a sequence or bag.
public static<E> void addAfter(Node<E> pre, E item) {
    if (pre != null) {
        pre.setLink(new Node<E>(item, pre.getLink( ));
    }
}

// 4b. Write a Bag method count which has the method header of
// public int count (E target)
// that will return an integer containing the number of times target is in the bag.
// Note that this is ugly because target can be null and cursor's data can be null
// I revised this after my first post
public int count (E target) {
    int times = 0;
    Node<E> cursor = head;
    while (cursor != null) {
        if (target == null) {
            if (cursor.getData( ) == null) {
                times++;
            }
        } else {
            if (target.equals(cursor.getData( ))) {
                times++;
            }
        }
        cursor = cursor.getLink( );
    }
    return times;
}

// 4c. Write a Bag method that has the method header of
// public Bag copyUnique ( )
// that returns a new bag that contains the same items as in the current Bag
// but only one copy of each unique item. No, you haven't seen this before.
// Realize that if the Bag you are creating meets its invariant, you may safely
// call its methods including insert and the count one in part 4b. Do not alter
// the existing Bag.
public Bag<E> copyUnique ( ) {
    Bag<E> newBag = new Bag<E>( );
    Node<E> cursor = head;
    while (cursor != null) {
        if (newBag.count(cursor.getData( )) == 0) {
            newBag.insert(cursor.getData( ));
        }
        cursor = cursor.getLink( );
    }
    return newBag;
}
public void add (E entry) {
    if (front == null) {
        front = new Node<E>(entry, null);
        rear = front;
    } else {
        rear.setLink(new Node<E>(entry, null));
        rear = rear.getLink();
    }
    manyNodes++; // if keeping a count of number of items