1. Given the IntNode class in your book,
   a. write an application-level method called printList prints out the contents of a linked list without changing the list. What should its parameter(s) be?
   b. write application-level code that will print out the value in the last node of a list called myList.
   c. Write a new static method for the IntNode class. It takes one parameter, which is a head node for a linked list. The method creates a new linked list, which is the same as the original list but in which all repetitions are removed. The method’s return value is a head reference for a new list.
   d. Write an application-level method with one parameter that is a head reference for a linked list, and creates a new list that has the same elements as the original list but in reverse order. The method returns a head reference for the new list.

2. Given the generic ArrayBag class in your book,
   a. Write application-level code to declare a new bag of type String.
   b. Write a method that accepts the bag you just created as a parameter, along with an integer \( \text{count} \), and reads \( \text{count} \) many strings into the bag.
   c. Write a new class method that removes all occurrences of a target value (passed as a parameter) from a bag. The method should return true if it removed at least one target value, and false if there were no occurrences of the target in the bag.

3. Explain why sometimes it is sufficient to use a test such as \( x == y \), while at other times you must write \( x.equals(y) \).

4. The clone method of a class such as ArrayBag does a “shallow copy”, which basically means that if the bag contains objects, the copying doesn’t go deep enough to make separate versions of each object. How might you create a “deep copy” clone operation, so that it would work with any type of bag, including ones that contain objects?

5. What is an interface in Java? How would it work for developing a MP3Player class, which implements the AudioClip interface?
6. Explain how the Lister class implements an iterator on a generic linked list. Write code to demonstrate the operations of Lister.

7. Show how you would evaluate the following postfix expressions using a stack, given $A = 1$, $B = 3$, and $C = -2$. Show all steps using the stack (similar to figure 6.11), not just the final picture.
   a. $A \cdot B + C - A \cdot$
   b. $B \cdot A \cdot C - /$
   c. $B \cdot C + A \cdot B \cdot C \cdot A + * -$

8. Add a method to the LinkedStack class called $\text{itemAt}$, that takes in an integer $n$ and returns the value of the item that is $n$ from the top (where the top is at $n = 0$). The stack may not be changed by the method. What should a reasonable precondition be? Can $\text{itemAt}$ throw an exception if the precondition isn’t met…if so, include that in your code.

9. Write an application-level method called $\text{compareStacks}$ that compares two stacks for equality, without changing either stack, returning the result of the comparison as a Boolean value.

10. Write an application-level method called $\text{totalQ}$ that accepts a queue of integers as a parameter and returns the total of all the values in the queue, without changing the queue.

11. Write a new accessor method for the linkedQueue class that returns a copy of the item at the rear of the queue.

12. Write a new method for the linkedQueue class called $\text{reverseQ}$ that returns a copy of a queue in reverse order.

13. Rewrite #12 as an application-level method instead.