Description
Problem 3.11 asked for a linked list class with only a reference to a header node, with 5 specific methods to implement.
The class has an inner Node private class to hold data and a reference to the next node. The class is given a private node variable for the head node.
The default constructor sets the head to -1. This acts as a “dummy” head, which is not used to store actual data, but rather to avoid complications in the code when dealing with the beginning of the list. The real head can still be accessed as head.next.
The size method starts at the head of the list, and traverses through it, updating a count variable with each node passed, until there are no nodes left in the list, after which it returns the number of items in the list.
The printList method also traverses through the list, printing out each element in the list with a space in between.
The contains method also traverses through the list, checking each element to see if it is equal to the parameter. If the end is reached without a match, the method returns false.
The add method also traverses through the list to check if the list already contains the element to be added, in which case the method is exited from. Otherwise, the element is added to the end of the list.
The remove method also traverses through the list until it finds the item to be deleted, after which it deletes the node by updating the next reference of the previous node. If the item is not found, the method does nothing.

A main class is included which tests every method in the LinkedList class.

Complexity Analysis
size: O(n), where n is the number of nodes in the list. Traverses the entire list to count the number of nodes.

printList: O(n). Traverses the entire list to print the data values of all nodes.

contains: O(n). Traverses the list to check if the value x exists in any node.

add: O(n). In the worst case, this method traverses the entire list to check if the value x already exists in any node, and then add a new node at the end of the list.

remove: O(n). In the worst case, this method traverses the entire list to find the node with value x and then remove it by updating the previous node's next reference.