Homework M5: Linked Node Implementations

Due Mar 7 at 11:59pm **Points** 100 **Questions** 26
Available until May 30 at 11:59pm **Time Limit** None

Instructions

Review the **Homework FAQ** page for details about submitting homework. In this homework, you will:

- evaluate node code
- work directly with node references
- implement methods to go inside the LinkedBag class
- implement methods to go inside the LList class

Homework Files

Below is the driver/tester program. I strongly recommend using this to test your code before submitting. You can ignore the *test methods* at the end of the file.

Important Note: The class headers have been updated to work with the driver program, so be sure to use this version of the files provided in the zip below (not from the lecture code).

HomeworkM5Files.zip (https://ccsf.instructure.com/courses/47904/files/7426748/download?download_frd=1)

Notes

Remember what it means to write from the implementation perspective.

- Writing from the implementation perspective means you are writing code to go **inside** the LinkedBag.java or LList.java class.
- Because you are writing code to go inside of these classes, you have access to the private instance data variables and private methods of the classes.
  - Often, accessing linked nodes directly allows you to write a much more efficient solution. It allows you to keep track of where you are in the node chain, rather than having to link through the chain over and over again.

Account for special cases.

- Unless explicitly stated otherwise, your code should run, produce an expected result, and **not crash** for an empty or singleton list, bag, or node chain.

- Reminder: for an empty bag/list, firstNode is null.
- Reminder: For a singleton bag/list, firstNode.next is null.
- You cannot invoke **anything** on null.

For full credit, think about efficiency.

- Your answer should be efficient.
- Sometimes, directly accessing instance data variables rather than invoking an existing method can yield a more efficient solution.
- Take advantage of writing from the implementation perspective by accessing the instance data variables when it will allow you to write a more efficient solution.
- For now, think of an efficient solution as one that **minimizes the number of passes through the chain of nodes and avoids nested loops when possible**. (We'll eventually learn how to truly measure efficiency!)
  - Often times, a nested loop will occur when a method involves going through the full chain and that method is then invoked inside of a loop that also goes through the full chain.
  - Calls to methods like getNodeAt, getReferenceTo, getEntry, contains, add (LList only) loop through the chain every time they are called. So calling these from ***within*** a loop that goes through the whole chain likely creates a nested loop and should be avoided.
  - You can often avoid this by directly accessing the instance data variables or invoking different methods.
- **To fully understand what I mean, make sure you have watched this week's Efficiency Example video (https://www.youtube.com/watch?v=mdJUO7FO_2I&list=PL5igFWijWBo3f_d53eNwkKBGfs1VaiLWt&index=17) .**

Do not use other kinds of data structures or invoke toArray.

- Do not invoke the toArray().
- Do not create a different data structure inside the code.
  - For example, do not create an ArrayList or LinkedList inside of a method in the LinkedBag class.
  - The goal of this homework is to practice with nodes and the LinkedBag and LList class- not with arrays or other classes.

This quiz was locked May 30 at 11:59pm.

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	3,262 minutes	103 out of 100

Score for this quiz: **103** out of 100
Submitted Mar 4 at 5:40pm
This attempt took 3,262 minutes.

Coding Questions

Question 1

16 / 16 pts

Write a method to print the values at a given interval in a chain of Strings, starting with the first node. The chain passed in should not be altered outside the method. You can see examples in the tester program.

The method header is:

```
public static void printAtInterval(Node<String> firstNo
de, int interval)
```

Your Answer:

```
// Question 1 homework 5
public static void printAtInterval(Node<String> firstNode, int interval) {
    // parameter
    int counter = 0;

    // alias firstNode
    Node<String> currentNode = firstNode;

    // output node data per interval number
    while (currentNode != null) {
        if ((currentNode != null) && (counter % interval == 0)) {
            System.out.print(currentNode.data + " ");
            currentNode = currentNode.next;
        } else if ((currentNode != null) && (counter % interval != 0)) {
            currentNode = currentNode.next;
        }
        // increment counter
        counter++;
    }
}
```

- The next questions ask you to write code at the developer level (or implementation perspective).
  - This means you are writing code that goes **inside** the LinkedBag and LList class.
- Read carefully to determine which ADT is being used.
- Carefully review the instance data variables and methods provided in those classes.
  - Use the private instance data variables (and sometimes private methods!) to create an efficient or improved solution.
- You will be graded both on syntax (the code should compile) and semantics (the code should accomplish the task).
- You will also be graded based on solution design; see the notes at the beginning of the homework for information about efficiency.

Question 2

16 / 16 pts

Write a removeAll method for the LinkedBag class. The method takes a value in as a parameter and removes **all** occurrences of that value. The method returns a count of how many elements were removed. The method header is:

```
public int removeAll(T element)
```

- Review the notes at the beginning of the homework about avoiding methods that result in nested loops.
- Invoking a method that loops through the chain from *inside of a loop* of the chain results in a nested loop.

Your Answer:

```
// Question 2 homework 5
public int removeAll(T item) {

    // counter
    int originalNumberOfEntries = numberOfEntries;
    int counter = 0;

    // alias to firstNode
    Node currentNode = firstNode;

    if (numberOfEntries == 0) { // checks for empty list
        return counter;
    } else if (numberOfEntries == 1 && firstNode.data.equals(item)) { // si
ngleton with match
        firstNode = firstNode.next;
        numberOfEntries--;
        counter++;
    } else if (numberOfEntries > 1) { // anything else
        // create temp bag
        LinkedBag<T> tempBag = new LinkedBag<T>();

        // empty currentBag items into tempBag
        while (currentNode != null) {
            tempBag.add(currentNode.data);
            currentNode = currentNode.next;
        }

        // reset the original temp bag
        this.clear();

        // add back in everything but the "item"
        for (int i = 0; i < originalNumberOfEntries; i++) {
            T itemClone = tempBag.remove();

            if (!itemClone.equals(item)) {
                this.add(itemClone);
            }
            counter = originalNumberOfEntries - numberOfEntries;
        }

    }
    return counter;
}
```

Question 3

16 / 16 pts

Write a getMax method for the LList class. The method returns the largest object in the list. The method header is:

```
public T getMax()
```

- Assume the list contains objects whose classes implement

Comparable.

- This means you can invoke compareTo on your objects of type T.
- The class header in LList has already been updated for this.

Your Answer:

```
// Question 3 homework 5
public T getMax() {
    // create alias for firstNode
    Node currentNode = firstNode;

    // assign max to data in first node
    T max = null;

    if (numberOfEntries == 0) { // for empty link-list
        return max;
    } else if (numberOfEntries == 1) { // return data in first node for sin
gleton
        max = currentNode.data;
        return max;
    } else { // else find max
        max = currentNode.data;

        while (currentNode != null) { // loop through link-list to compare
data
            if (currentNode.data.compareTo(max) > 0) {
                max = currentNode.data;
            }
            // transverse link-list
            currentNode = currentNode.next;
        }
    }
    return max;
}
```

Question 4

7 / 10 pts

Write an addAll method for the LList class. The method adds an array of items (in order) to the **end** of the current list. The method header is:

```
public void addAll(T[] items)
```

- Review the notes at the beginning of the homework about avoiding methods that result in nested loops. Invoking a method that loops through the chain from *inside of a loop* of the chain results in a nested loop.

Your Answer:

```
// Question 4 homework 5
public void addAll(T[] items) {
    // create alias for firstNode
    Node currentNode = firstNode;

    if (numberOfEntries == 0 || items.length == 0) { // if this or other li
nked-list is empty
        // just add in new items if empty list
        for (int i = 0; i < items.length; i++) {
            this.add(items[i]);
        }

    } else if (numberOfEntries == 1) { // return data in first node for sin
gleton
        // store original one item in temp variable
        T tempData = currentNode.data;

        // loop to add new items
        for (int i = 0; i < items.length; i++) {
            this.add(items[i]);
        }
        // add back on original data
        this.add(tempData);

    } else if (numberOfEntries > 1) {
        // create a temp link-list
        LList<T> tempNode = new LList<T>();

        // copy original link-list into a temp link-list
        while (currentNode.next != null) { // loop through link-list to emp
ty data into temp link-list
            tempNode.add(currentNode.data);
            currentNode = currentNode.next;
        }

        // loop to add new items to linked-list
        for (int i = 0; i < items.length; i++) {
            this.add(items[i]);
        }

        // loop to add back original items
        for (int i = 1; i < tempNode.numberOfEntries - 1; i++) {
            this.add(tempNode.getEntry(i));
        }
    }
}
```

-3 repeated calls to add(T) within a loop is very inefficient; each call to add has to traverse the entire chain, so this happens *each* time you add an element from the array; instead, take advantage of access to the node structure to only link through the chain one time and then save a reference to that last node and manually create new nodes to link up to the end o the chain; this way, you only have to iterate the chain once

Multiple Choice Questions

The next set of questions show code snippets and ask you to describe the contents of the chain referenced by firstNode and the value of the data in the node called currentNode.

```
Node a = new Node(5);
Node b = new Node(3, a);
Node c = new Node(7, b);
Node d = new Node(2, c);
Node firstNode = d;
Node currentNode = firstNode.next;
```

Question 5

2 / 2 pts

What is the contents of the chain referenced by firstNode?

Correct!

- (•) 2->7->3->5
- ( ) 5->7->3->2
- ( ) none of these is correct
- ( ) 7->3->2

- 5
- 3->7->5
- 2

Question 6

2 / 2 pts

What is the data held in the node referred to by currentNode?

- 2
- none of these is correct

Correct!

- (selected) 7
- 5
- 3

```
Node a = new Node(5);
Node b = new Node(3, a);
Node c = new Node(7, b);
Node d = new Node(2, c);
Node firstNode = d;
Node currentNode = firstNode;
currentNode.data = 15;
```

Question 7

2 / 2 pts

What is the contents of the chain referenced by firstNode?

- 2->15->3->5
- **Correct!** 15->7->3->5
- none of these is correct
- 2->7->3->5
- 2
- 15

Question 8

2 / 2 pts

What is the data held in the node referred to by currentNode?

- none of these is correct
- 7
- 3
- 5
- 2
- **Correct!** 15

```
Node a = new Node(5);
```

```
Node b = new Node(3, a);
Node c = new Node(7, b);
Node d = new Node(2, c);
Node firstNode = d;
Node currentNode = firstNode;
```

Question 9

2 / 2 pts

What is the contents of the chain referenced by firstNode?

- 7 -> 3 -> 5
- 2 -> 7 -> 3 -> 5 **Correct!**
- 3 -> 7 -> 2
- none of these is correct
- 5 -> 3 -> 7 -> 2

Question 10

2 / 2 pts

What is the data held in the node referred to by currentNode?

- 3
- 5
- 2 **Correct!**
- 7

- none of these is correct

```
Node a = new Node(5);
Node b = new Node(3, a);
Node c = new Node(7, b);
Node d = new Node(2, c);
Node firstNode = d;
Node currentNode = firstNode;
currentNode = currentNode.next;
```

Question 11

2 / 2 pts

What is the contents of the chain referenced by firstNode?

- none of these is correct
- 7

Correct!

- 2->7->3->5
- 7->7->3->5
- 2
- 7->3->5

Question 12

2 / 2 pts

What is the data held in the node referred to by currentNode?

- 3
- none of these is correct
- 5
- 7 **Correct!**
- 2

```
Node a = new Node(5);
Node b = new Node(3, a);
Node c = new Node(7, b);
Node d = new Node(2, c);
Node firstNode = d;
Node currentNode = firstNode;
currentNode.next.next.data = 8;
```

Question 13

2 / 2 pts

What is the contents of the chain referenced by firstNode?

- 2->7
- 2->7->5
- 2->7->8->5 **Correct!**

- none of these is correct
- 2->7->3->5
- 8->5

Question 14

2 / 2 pts

What is the data held in the node referred to by currentNode?

- 5
- none of these is correct

Correct!

- 2
- 8
- 7
- 3

```
Node a = new Node(5);
Node b = new Node(3, a);
Node c = new Node(7, b);
Node d = new Node(2, c);
Node firstNode = d;
Node currentNode = firstNode;
firstNode = firstNode.next;
```

Question 15

2 / 2 pts

What is the contents of the chain referenced by firstNode?

- 5 -> 7 -> 2
- 2 -> 7 -> 3 -> 5
- **Correct!** 7 -> 3 -> 5
- 5 -> 3 -> 7 -> 2
- 2 -> 3 -> 5
- none of these is correct

Question 16

0 / 2 pts

What is the data held in the node referred to by currentNode?

- 5
- **ou Answered** 7
- none of these is correct
- 3
- **orrect Answer** 2

```
Node a = new Node(5);
Node b = new Node(3, a);
Node c = new Node(7, b);
Node d = new Node(2, c);
Node firstNode = d;
Node currentNode = firstNode;
currentNode.next = currentNode.next.next;
```

Question 17

2 / 2 pts

What is the contents of the chain referenced by firstNode?

- 2->7->3->5
- 2->3->3->5
- 7->3->5
- none of these is correct
- 2
- 2->7->5
- 2->5->3->5

Correct!

- (selected) 2->3->5
- 3->5
- 2->5

Question 18

2 / 2 pts

What is the data held in the node referred to by currentNode?

- none of these is correct
- 7
- 3
- 5
- 2 (selected)

Correct!

```
Node a = new Node(5);
Node b = new Node(3, a);
Node c = new Node(7, b);
Node d = new Node(2, c);
Node firstNode = d;
Node e = new Node(6);
e.next = firstNode.next;
firstNode.next = e;
```

Question 19

2 / 2 pts

What is the contents of the chain referenced by firstNode?

- none of these is correct
- 2 -> 7 -> 6 -> 3 -> 5

Correct!

- (selected) 2 -> 6 -> 7 -> 3 -> 5
- 2 -> 6
- 6 -> 2 -> 7 -> 3 -> 5
- 6 -> 7

```
Node a = new Node(5);
Node b = new Node(3, a);
Node c = new Node(7, b);
Node d = new Node(2, c);
Node firstNode = d;
firstNode.next.next = firstNode.next.next.next;
```

Question 20

2 / 2 pts

What is the contents of the chain referenced by firstNode?

Correct!

- (selected) 2 -> 7 -> 5
- 2 -> 7 -> 3
- a NullPointerException will be thrown
- 2 -> 7 -> 3 -> 5
- none of these is correct
- 7 -> 3 -> 5

2 -> 3 -> 5

```
Node a = new Node(5);
Node b = new Node(3, a);
Node c = new Node(2, b);
Node firstNode = c;
Node currentNode = firstNode;
currentNode = currentNode.next;
currentNode = currentNode.next;
System.out.println(currentNode.next);
```

Question 21

2 / 2 pts

What is the result of the println?

- A 3 is printed
- A 5 is printed
- none of these is correct

Correct!

- A null is printed
- An exception/error is thrown
- A 2 is printed

```
Node a = new Node(5);
Node b = new Node(3, a);
```

```
Node c = new Node(2, b);
Node firstNode = c;
Node currentNode = firstNode;
currentNode = currentNode.next;
currentNode = currentNode.next;
System.out.println(currentNode.next.data);
```

Question 22

2 / 2 pts

What is the result of the println?

- A 2 is printed
- An exception/error is thrown

Correct!

- A null is printed
- none of these is correct
- A 3 is printed
- A 5 is printed

Question 23

2 / 2 pts

Assume you have a linked list that contains 4 -> 6 -> 8 -> 9, with a firstNode reference to the first node (the node with the 4 data). What will print?

```
Node currentNode = firstNode;

while(currentNode.next != null) {
   System.out.print(currentNode.data);
```

```
    currentNode = currentNode.next;
}
```

Correct!

- [x] 468
- [ ] 4689
- [ ] 689
- [ ] 68
- [ ] none of these is correct

Question 24

2 / 2 pts

What is true of the following method?

```
public void method(Node firstNode) {
   while(firstNode != null) {
      System.out.println(firstNode.data);
      firstNode = firstNode.next;
   }
}
```

Correct!

- [x] none of these is correct
- [ ] it will crash on an empty list but not on a singleton list
- [ ] it will crash on an empty list and singleton list
- [ ] there will be an infinite loop

Question 25

0 / 2 pts

What is true of the following method?

```
public void method(Node firstNode) {
   Node secondNode = firstNode.next;
   Node currentNode = secondNode;

   while(currentNode != null) {
      System.out.println(currentNode.data);
      currentNode = currentNode.next;
   }
}
```

orrect Answer

- it will crash on an empty list but not on a singleton list

- it will crash on an empty list and a singleton list

ou Answered

- none of these is correct

- there will be an infinite loop

Optional Extra Credit: 10 points

Write a method to determine if a doubly-linked list is a "palindrome list"- meaning that the list contains equivalent elements front-to-back and back-to-front.

The method header is:

```
public static boolean isDoublyLinkedPalindromeList(Doub
leNode<Integer> firstNode, DoubleNode<Integer> lastNod
e)
```

- firstNode is the first node in the list and lastNode is the last node **in the same list**.

- For full credit, do not create a new data structure. Use only the references to the DoubleNode parameters.
- Examples:
  - 1, 2, 3, 2, 1 is a palindrome list
  - 1, 2, 2, 1 is a palindrome list
  - 1, 2 is not a palindrome list
  - 1, 2, 3, 4, 5 is not a palindrome list

Question 26

10 / 0 pts

Paste your complete method here.

Your Answer:

```
// Question 26 homework 6
public static boolean isDoublyLinkedPalindromeList(DoubleNode<Integer> firs
tNode, DoubleNode<Integer> lastNode) {
    // assigning alias to firstNode and lastNode
    DoubleNode<Integer> headNode = firstNode;
    DoubleNode<Integer> tailNode = lastNode;

    // loop to catch when not a palindrome
    if (headNode != null && tailNode != null) {
        if (headNode.data != tailNode.data) {
            return false;
        }
        while (headNode.next != null && tailNode.previous != null) {
            headNode = headNode.next;
            tailNode = tailNode.previous;
            if (headNode.data != tailNode.data) {
                return false;
            }
        }
    }
    return true;
}
```

note that it's best to always use the equals method, not ==, to compare objects for logical equivalence; even for Integer objects! for Integers, you can also convert to int using integerObject.intValue() and then safely use ==

Quiz Score: **103** out of 100