CSCI 2110 Data Structures and Algorithms
Fall 2024
Laboratory No. 5
Week of October 14 – October 18

Due (on Brightspace): Sunday, October 20, 11.59 PM

<u>Linked Lists</u>

Note: Monday, October 14th is a holiday. There will be no lab session on this day. Students from Monday's lab can attend any other session during the week.

In this lab, you will continue your experimentation with algorithm complexity. Your task is to write, compile and run each program using an Integrated Development Environment (IDE). You may use any IDE of your choice – IntelliJ, Eclipse, NetBeans, VSCode, etc.

<u>Marking Scheme</u>: Throughout this course, we will be focusing on the complexity of algorithms and consequently, the efficiency of programs. In your first-year courses, the focus was on creating runnable code to specifications. This course builds on previous knowledge, but also focuses on efficiency. This will involve reducing unnecessary coding and designing or choosing good algorithms.

Each exercise will list the test cases that you need to test the code. Ensure that the output that you generate covers all the test cases.
Each exercise carries 10 points.
Your final score will be scaled down to a value out of 10. For example, if there are three exercises and you score 9/10, 10/10 and 8/10 on the three exercises, your total is 27/30 or 9/10.

Commenting your code: Your code should be clear and easy to understand. While extensive comments are not required, you are expected to include <u>brief, meaningful comments</u> to explain complex logic or any non-obvious sections of your code. Use inline comments to clarify key parts of the code. For methods and classes, use Javadoc to briefly describe their purpose, parameters, and return values. This ensures others can understand your code's intent quickly.

Error checking: Unless otherwise specified, you may assume that the user enters the correct data types and the correct number of input entries, that is, you need not check for errors on input.

Suggested coding practice: Don't try to tackle the entire exercise in one go. Code a little, test a little, code a little, test a little. You will have better progress if you frequently compile and test your code (every 5-10 lines).

Submission: One zip file containing the source codes (.java files) for all classes, and sample outputs. Please see details at the end of this document.

Submission Deadline: Sunday, October 20, 2024, 11.59 PM

Grace Time: Submissions will be accepted until 4.59 AM on Monday, October 21, 2024 without late penalty.

Late Penalty: Submissions received after the grace time will be subject to a 10% per day late penalty. The submission portal will close on Wednesday, October 23, 4.59 AM.

Dropping of Labs: Up to 2 labs can be dropped during the semester. No SDA submission is required.

Exercise

There is only one exercise for this lab. It requires you to write a demo program with a main method that creates and manipulates a linked list of Strings. You will be required to add methods to the demo program.

First download the Node.java and LinkedList.java files provided along with this lab document.

1. The program should first prompt the user to enter a list of Pokemon names (Strings), create a linked list with the Pokemon names and display the list, as shown in the example below:

```
Enter Pokemon names, one on each line
Enter quit to end
Pikachu
Charizard
Mewtwo
Emboar
Bulbasaur
Eevee
Gengar
Ditto
Charmander
quit

Here is the linked list:
  Charmander-->Ditto-->Gengar-->Eevee-->Bulbasaur-->Emboar-->Mewtwo-->Charizard-->Pikachu-->
```

2. Next add a method to the demo program that accepts a linked list of Strings as the argument and displays the names in the nodes in even-numbered indices in the list. Assume that the first node is at index 0. For example, with the above list, it should display

```
Here is the linked list with even-numbered indices:
Charmander  Gengar          Bulbasaur   Mewtwo          Pikachu
```

Note that your method should work for a linked list of any size.

3. Next add a method to the demo program that that accepts a linked list of Strings as the argument and displays the names in the nodes in odd-numbered indices in the list. Assume that the first node is at index 0. For example, with the above list, it should display

```
Here is the linked list with odd-numbered indices:
Ditto       Eevee       Emboar          Charizard
```

Note that your method should work for a linked list of any size.

4. Next add a method to the demo program that that accepts a linked list of Strings as the argument and creates and returns another linked list with the names in the reverse order. For example, if the linked list that you created above is the argument, it should create and return the following list:

```
Here is the reversed linked list:
```

Pikachu-->Charizard-->Mewtwo-->Emboar-->Bulbasaur-->Eevee-->Gengar-->Ditto-->Charmander-->

Note that your method should work for a linked list of any size.

5. Next add a method to the demo program that removes the middle node in the linked list. For example, if the list is

Charmander-->Ditto-->Gengar-->Eevee-->Bulbasaur-->Emboar-->Mewtwo-->Charizard-->Pikachu-->

then it should remove the node at index 4 (“Bulbasaur” - midpoint between 0 and 8) and change the list to

Charmander-->Ditto-->Gengar-->Eevee-->Emboar-->Mewtwo-->Charizard-->Pikachu-->

Suppose the list has an even number of nodes. For example, the list below has 8 nodes,

Charmander-->Ditto-->Gengar-->Eevee-->Emboar-->Mewtwo-->Charizard-->Pikachu-->

then the method should remove the node at index 3 (“Eevee”, since 7/2 in integer division is 3), and change the list to

Charmander-->Ditto-->Gengar-->Emboar-->Mewtwo-->Charizard-->Pikachu-->

Your method should work for a linked list of any size.

Test your program by running it for at least three different sets of inputs.

Here’s the structure of your solution:

```
import java.util.*;
public class LinkedListDemo{
     public static void main(String[] args){
            //continue
     }
     //method to display even-numbered nodes
     public static void displayEven(LinkedList<String> list){
     //continue
     }
     //method to display odd-numbered nodes
     public static void displayOdd(LinkedList<String> list){
     //continue
     }
     //method to create a reversed linked list
     public static LinkedList<String> reverse(LinkedList<String> list){
     //continue
     }

     //method to remove the middle node in the linked list
     public static void removeMiddle(LinkedList<String> list){
     //continue
     }
}
```

Submission items

One zip file containing the following:

1. Node.java → This is the unchanged Node.java code that was provided to you.
2. LinkedList.java → This is the unchanged Node.java code that was provided to you.
3. LinkedListDemo.java → This is your code
3. Text/word document containing screen capture of three sample runs of the program.

At the top of LinkedListDemo.java file, put the following information in comments.

// Full Name: [Your Full Name]
// ID Number: [Your Banner ID]

You MUST SUBMIT .java files that are readable by the TA. If you submit files that are unreadable such as .class, you will lose points.