LAB #9 - LINKED LISTS

PART 1: Exercises

1. Use the Java code below to draw the diagram and identify the values of the variables/expressions that follow. The value may be undefined, or the expression may be invalid. **NOTE:** For this exercise, each node has 3 components (`number-character-link`).

```
LinkedListNode current = new LinkedListNode();
LinkedListNode last = new LinkedListNode();

current.number = 37;
current.character = 'z';
current.link = new LinkedListNode();
last = current.link;
last.number = 9;
LinkedListNode first = new LinkedListNode();
last.link = first;
first.number = 9;
first.character = 'h';
first.link = current;
```

Expression:

```
1) first.link.number
2) first.link.link.character
3) first.link == last
4) current.link.number
5) first == last.link
6) first.number < first.link.number
```

---

2. Use the Java code below to draw the diagram and show the output. **NOTE:** For this exercise, each node has the usual 2 components (`info-link`).

```
LinkedListNode current = new LinkedListNode();
current.info = 10;
LinkedListNode node = new LinkedListNode();
node.info = 27;
node.link = null;
current.link = node;
node = new LinkedListNode();
node.info = 20;
node.link = current.link;
current.link = node;
System.out.println(current.info + "  " + node.info);
node = node.link;
System.out.println(node.info);
```

---

3. Use the Java code below to draw the diagram and show the output. **NOTE:** For this exercise, each

node has the usual 2 components (`info-link`) .

```
LinkedListNode current = new LinkedListNode();
current.info = 10;
LinkedListNode node = new LinkedListNode();
node.info = 27;
node.link = null;
current.link = node;
node = new LinkedListNode();
node.info = 20;
node.link = current;
current = node;
node = new LinkedListNode();
node.info = 37;
node.link = current.link;
current.link = node;
node = current;
while(node != null) {
    System.out.println(node.info + "  ");
    node = node.link;
}
```

PART 2: Programming

1. Write the implementation for a linked list of integers (modify/adapt for `int` the generic implementation discussed in class). Have the following:

```
//Interface: LinkedListIntADT
public interface LinkedListIntADT {
    public boolean isEmptyList();
    public void initializeList();
    public void print();
    public int length();
    public int front();
    public int back();
    public boolean search(int searchItem);
    public void insertFirst(int newItem);
    public void insertLast(int newItem);
    public void deleteNode(int deleteItem);
}

//Class: LinkedListIntClass implements
//Interface: LinkedListIntADT
import java.util.*;
public abstract class LinkedListIntClass implements LinkedListIntADT {
    ...
}

//Class: UnorderedLinkedListInt extends
//Class: LinkedListIntClass
public class UnorderedLinkedListInt extends LinkedListIntClass {

    ...
```

```
    }
```

---

2. Add to the class `UnorderedLinkedListInt` a value-returning member method named `findSum` that returns the sum of all the data values in a list. Work with the above list of integers (`int`).

---

3. Add to the class `UnorderedLinkedListInt` a value-returning member method named `findMin` that returns the smallest of all the data values in a list. Work with the above list of integers (`int`).

---

4. Add to the class `UnorderedLinkedListInt` a `toString` method to create a comma-separated, bracketed version of the list (as in the sample output below). Work with the above list of integers (`int`).

---

5. Test the new methods using the client below. Handle input validation.

```
//Class: ClientUnorderedLinkedListInt
//Input: 37 10 88 59 27 20 14 32 89 100 12 999
import java.util.*;
public class ClientUnorderedLinkedListInt {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        UnorderedLinkedListInt intList = new UnorderedLinkedListInt();
        UnorderedLinkedListInt tempList;
        int num;
        System.out.println("Enter integers (999 to stop)");
        num = input.nextInt();//valid??
        while (num != 999) {
            intList.insertLast((Integer) num);
            num = input.nextInt();//valid??
        }
        System.out.print("\nTesting .insertLast and .print. The original list is:
");
        intList.print();
        System.out.println("\nTesting .length. The length of the list is: " +
intList.length());
        if (!intList.isEmptyList()) {
            System.out.println("Testing .front. First element/list: " +
intList.front());
            System.out.println("Testing .back. Last element/list: "  +
intList.back());
        }
        System.out.println("Testing .sum. The sum of data in all nodes is: " +
intList.findSum());
        System.out.println("Testing .min. The smallest data in all nodes is: " +
intList.findMin());
        System.out.print("Testing .search. Enter the number to search for/list: ");
        num = input.nextInt(); //valid??
        if (intList.search(num))
            System.out.println(num + " found in this list.");
        else
            System.out.println(num + " is not in this list.");
        System.out.print("Testing .remove. Enter the number to be deleted from list:
```

```
");
        num = input.nextInt();//valid??
        intList.deleteNode(num);
        System.out.print("Testing .toString. After deleting " + num + ", the list
is: " + intList);
        System.out.println("\nThe length of the list after delete is: " +
intList.length());
        //Optional: add more testing here
    } // add methods for input validation
}
```

OUTPUT:

```
Enter integers (999 to stop)
37 10 88 59 27 20 14 32 89 100 12 999
Testing .insertLast and .print. The original list is: 37 10 88 59 27 20 14 32 89 100
12
Testing .length. The length of the list is: 11
Testing .front. First element/list: 37
Testing .back. Last element/list: 12
Testing .sum. The sum of data in all nodes is: 488
Testing .min. The smallest data in all nodes is: 10
Testing .search. Enter the number to search for/list: 20
20 found in this list.
Testing .remove. Enter the number to be deleted from list: 59
Testing .toString. After deleting 59, the list is: [37, 10, 88, 27, 20, 14, 32, 89,
100, 12]
The length of the list after delete is: 10
```
