C S 272/463 Introduction to Data Structures

Lab 4: Singly Linked List - Basic Operations

I. Requirements

1. Design and implement the `IntNode` class and put it to IntNode.java with the following detailed requirements.

   (1) (5 pts) This class has two instance variables:

   One instance variable is for keeping an integer value, and the other instance variable is a link pointing to another `IntNode` instance or pointing to null.

   You should NOT add any new instance variables.

   (2) (5 pts) The no-argument constructor which sets the node value to be 0 and the link to be null reference.

   ```
   public IntNode()
   ```

   (3) (5 pts) A constructor with the given node value and the link.

   ```
   public IntNode(int _data, IntNode _node)
   ```

   (4) (10 pts) Get and set methods to get the node value and node link.

   (5) (10 pts) toString method

   ```
   public String toString()
   ```

   This method should return a `String` for the linked list starting from the node that activates this method. E.g., if the head node of Figure 1 activates this method, the output should be

   12->28->0->34

   If the third node of Figure 1 activates this method, it should output

   0->34

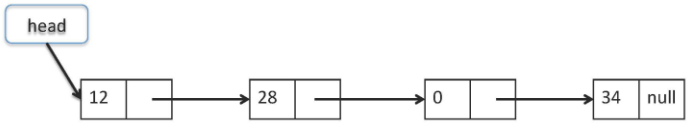

Figure 1: An example linked list

   (6) (7 pts) A method to add a node after the current node.

   ```
   public void addNodeAfterThis(int newdata)
   ```

   This method should create a new node with value newdata and let the current node's link point to this new node.

   For instance, if the current node contains content 5 and its link points to another node with content 10.

   5->10

   Then, activating `addNodeAfterThis(20)` from the node with content 5 will generate a new list

   5->20->10

(7) (8 pts) A method to remove the node after the current node.

```
public void removeNodeAfterThis()
```

This method should remove the node that this node's link points to.

(8) (12 pts) A method to get the number of nodes in the list starting from a given node head.

```
public static int listLength(IntNode head)
```

(9) (13 pts) A method to search whether a linked list starting with head contains a given value data.

```
public static boolean search(IntNode head, int data)
```

This method returns true if data exists in the linked list starting with head; It returns false otherwise.

Precondition of this method is that head is not null.

(10) (10 pts) A method to find the *m*th-to-last element of the list.

Implement your algorithm, taking care to handle relevant error conditions. Define *m*th to last such that *m*=0, the last element of the list is returned.

```
public IntNode findMthToLast IntNode header, int m)
```

2. (15 pts) Implement IntNodeTest.java to test all the methods in IntNode.java.

   Implement a `main()` method to thoroughly test all the methods in IntNode.java. Design test cases, put them in your main method, run your program through the test cases.

II. Note

- Specifications for all your classes and methods:

  Please properly explain (1) the functionality of the methods, (2) the parameters, (3) the return values, (4) the pre-conditions if there is any;

  Please use inline comments, meaningful variable names, indentation, formatting, and whitespace throughout your program to improve its readability.

- You can (but are not required to) design and implement other facilitating methods (E.g., other get and set methods, toString method) to finish the implementation of the required methods.

III. Submission

Submit through canvas a zipped file containing your (1) java file(s) (not .class files)

IV. Grading Criteria

1. The score allocation is beside the questions.
2. Please make sure that you test your code thoroughly by considering all possible test cases. Your code may be tested using more test cases.