C S 272/463 Introduction to Data Structures

Lab 5: Singly Linked List (advanced)

I. Requirements

1. Add the following methods to IntNode.java that you implemented in the previous lab. Note that all the methods except the hasCycle method assume that the given linked list does not have cycles.

   (1) (10 pts) A method to test whether a linked list starting from the given head is cyclic or acyclic. Your function should return true if it is cyclic. Otherwise, it should return false if the list is acyclic. NOTE: You should NOT modify the list content in any way.

   ```
   public static boolean hasCycle(IntNode head)
   ```

   (2) (10 pts) A method to change the node value at index `i`. Note that node `head` has index 0.

   ```
   public static void changeElement(IntNode head, int i, int newValue)
   ```

   (3) (20 pts) Write a method to modify the linked list such that all even numbers appear before all the odd numbers in the modified linked list. Also, keep the order of even and odd numbers same. If all numbers are even then do not change the list. If all numbers are odd then do not change the list.

   Input: 8->7->10->5->4->1->6->NULL

   Output: 8->10->4->6->7->5->1->NULL

   ```
   public static void segregate_even_odd(IntNode head)
   ```

   (4) (20 pts) A method to reverse a linked list.

   ```
   public static IntNode reverse (IntNode head)
   ```

   This method should return the linked list with the new head.

   For example, given a list

   12->28->0->34

   - If the input parameter head points to the node with value 12, this function should return

     34->0->28->12.

   - If the input parameter head points to the node with value 28 (which implicitly passes the list 28->0->34), this function should return 34->0->28. Note that 12 should not be shown in the reversed list.
   - If the input parameter head points to the node with value 0 (which implicitly passes the list 0->34), this function should return 34->0. Note that 12 and 28 should not be shown in the reversed list.

   NOTE: the elements in the list starting from head do not need to be ordered.

   (5) (10 pts) A method to swap the current node with the last node.

   ```
   public void swap()
   ```

   For example, given a list

   12->28->0->34

   If the current node that activates the method is 28, this method should return 12->34->0->28

(6) (20 pts) Given a singly linked list, rotate the linked list counter-clockwise by *k* nodes. Where *k* is a given positive integer. For example, if the given linked list is 10->20->30->40->50->60 and *k* is 4, the list should be modified to 50->60->10->20->30->40. Assume that *k* is smaller than the count of nodes in a linked list.

```
public static IntNode rotate(IntNode head)
```

This method should return the linked list with the new head.

2. (10 pts) Implement IntNodeAdvancedTest.java with test cases to test all the above 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.