C S 272/463 Introduction to data structures
Fall 2019
Lab 5: Singly Linked List (advanced)

1 Learning objectives

Objective 1 (list), Objective 5, Objective 6, Objective 7

2 Requirements

- 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 calculate the number of even elements in the linked list staring from the given `head`. When `head` is `null`, return 0.

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

  2 (10 pts) A method to add the given `newdata` to the end of the linked list that starts from the current node.
  NOTE: the elements in the list starting from the current node do not need to be ordered.

  ```
  public void addToEnd(int newdata)
  ```

  3 (10 pts) The following method to calculate the summation of elements in the last `num` nodes in a given linked list.

  ```
  public static int sumLast(IntNode head, int num)
  ```

  When `num` is bigger than the number of nodes in the list, it should return the summation of elements in all the nodes. For example, given the following list and let head point to its first node.

  ```
  12->28->12->28
  ```

  `sumLast(head, 1)` should return 28;
  `sumLast(head, 2)` should return 40 (=28+12);
  `sumLast(head, 5)` should return 80 (=28+12+28+12);
  Please put proper precondition.

  4 (15 pts) Copy part of a given linked list.

  ```
  public static IntNode copyOdd (IntNode head)
  ```

  If `head` is `null`, this method returns `null`.
  If `head` is not `null`, this method should copy all the odd elements in the linked list starting from the given head, create a new linked list with all these odd numbers, and return the linked list with the new head.

  5 (15 pts) A method to remove ALL the nodes that have the data value `e` from the linked list starting from the given `head`. This method should return the linked list with the new head. If `head` is `null`, this method returns `null`.

  ```
  public static IntNode removeAll(IntNode head, int e)
  ```

  NOTE: you are NOT allowed to create more than two `IntNode` objects. You can use `IntNode` references.
  The elements in the list starting from head do not need to be ordered.

  6 (15 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.

7 (15 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)
```

- (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.

3 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.

4 Submission

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

5 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.