**COP 3330 Spring 2024**

**University of Central Florida**

Assignment3

GUIDELINES

Your all source codes have the statement at the beginning.

package assignment3;

In other words all your source code will be in the same package. Do not use default packaging.

You cannot use Collections in Java for this assignment.

You cannot use any other 3rd party library.

Your code should compile fine. Non compiling submissions will receive a penalty.

**Shrinking Linked List Assignment Clarification**

**General Rules**

This assignment is about linked lists in Java. You will implement a linked list and do operations on that.Your linked list implementation will not have a *setItem* method. Therefore changing the element inside the linked list node after it has been created, will NOT be possible. You will lose points if you add a method to modify the linked list node content. *You can not store the elements in an array*! You also cannot use doubly linked list for this assignment.

You will need to implement a *Node* class which does not have setItem method

You will need to implement a *LinkedList* class.

You will need to implement a *LinkedListTester* class which will have a main method inside.

The file names will be as specified above. If you use a different file name, autograder will have issues with your assignment. You will lose points.

**Clarification**

In this assignment you will read a data file named as data.txt. The filename will be hardcoded.

This data file contains integers with a space between them. You will add the numbers which are **greater than zero** to a linked list. Then you will traverse this linked list from the beginning and delete some nodes and add a new one. The rules are below

when start traversing the linked list, you will need to store *the count*  number to some variable. It will be *-100* at the beginning. While traversing If you see this condition in the list,

X[i]=X[i-1] x 2 + 7

You will delete X[i] and X[i-1] and add a node that contains the item *the count*. You will increase *the count* number as you make insertions to the list. After you have done this you can continue traversing. You won’t need to check the previous element after adding a new node. When you reach at the end of the list you will add a new node with *the count* number as the item.

An example data.txt and processed.txt is added to this document

Example:

If your linked list contain those numbers:

1 4 3 9 25 6 7 21 8 9 19

|

current

here 9 x2 + 7 =25

When we reach the element 9

You will delete 9 and 25 and add -100

After insertion our current node will be -100

previous will be 3

next one will be 6

*Note that this shrinking will not happen again once a shrink has taken place. We will not use such data to test your software.*

So the linked list will become like this:

1 4 3 -100 6 7 21 8 9 19

At the next iteration:

1 4 3 -100 6 7 21 8 9 19

|

current

when we continue and reach 7 we see that 7 x 2 + 7=21. You will delete 7 and 21 and add a node that contains -99.

After insertion our current node will be -99

previous will be 6

next one will be 8

The linked list becomes

1 4 3 -100 6 -99 8 9 19

Until the end we won’t see the condition again. Therefore at the end of the list we will add -98. This addition will make sure that the array is processed.

the last linked list becomes

1 4 3 -100 6 -99 8 9 19 -98

You will need to write a text file called *processed.txt* which contains the linked list elements. The items will have a space between them.

Even you print the linked list to the output console correctly, you will not receive points for output.

data.txt

-1 5 1 -5 4 3 -4 9 -7 -88 25 17 7 -25 21 7 9 19 -99 -87

processed.txt

5 1 4 3 -100 17 -99 7 9 19 -98