**COP 3330**

**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. Do not use ArrayList or Array to implement this assignment. You need to use LinkedList.

Do not use static imports.

All instance variables should be private and all methods should be public.

Do not provide absolute file path. just use “processed.txt” and “data.txt”. If your path includes some folders which is specific to your local machine, the output will not be created. You will lose considerable amount of points if you assignment does not create output in our system.

You cannot use any other 3rd party library.

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

Do not submit class files. Please submit a zip file which contains java files.

**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 Node 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.

You can implement any LinkedList method as you prefer, however, the two methods below is necessary.

public void construct(String fname)

This method will construct a linked list by reading the file. The constructed linked list will have no negative elements inside.

public void process()

This method will process the link list based on the description below.

**Clarification**

In this assignment you will read a data file named as data.txt. The filename will be hardcoded. The data file will not be empty. 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. *count* will be initialized to -100. 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 since the added element will be negative. In other words, once a shrink has happened, it cannot happen again with the next element. 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.*

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 single space between them. There is no space at the end of the output in the processed.txt file.

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

Another test case

**data.txt**

15 37 -20 7 -3 60 127 31 69 11 9 -5 20 74 155 22 21 69 145 9 6 4 -16 14 -12 36 79 8 14 17 41 -23 11 5 22 -18 -17 82 171 18 27 61 20 -11 49 105 -7 24 -4 80 167 35 77 23 4 4 15 -20 -13 21 8 -7 -15 -23 -22 14 -18 -7 5 0 20 -7 -4 9 25 -20

**processed.txt**

-100 7 -99 -98 11 9 20 -97 22 21 -96 9 6 4 14 -95 8 14 -94 11 5 22 -93 18 -92 20 -91 24 -90 -89 23 4 -88 21 8 14 5 20 -87

Note that if the last two elements are processed and a negative number is added, we don’t add another negative number at the end.

It is also possible that the first two elements should shrink.