**Linked List Program**

Situation: Terrance Torrance presented a model on Creativity Analysis in which one portion describes “Unique Word Count” – a concept also used in the speller-module of any good word processor. In this program, you are going to write a program to do a creativity analysis based on the unique word count.

Required parameters:

* A file containing a paragraph…
* A real paragraph with upper and lower case characters and punctuation
* All words may be assumed to be spelled correctly
* Start with a tiny paragraph but make sure it works on a good sized one, too.
* No hyphenated words (for simplicity)
* White space shall be defined (for this program) as a blank space, a blank and its preceding punctuation, and non-printable characters (like \n)
* An output file reporting on the creativity of the paragraph
* No direct contact with the user – the user will simply run the program with the name of the file as a runtime parameter and you will create the report.

Program details:

* Create a Word class that contains a word (string) and a count (int) plus getters/setters.
* Create a Node class and a LinkedList class (see ppt for specifications on these)
  + STRONG SUGGESTION: Test your linked list class before moving on by writing a testing program that creates a linked list then calls each method one at a time, printing after each call.
* Read the paragraph
* Create a list of unique words by creating a linked list of Word objects
* Count how often each word appears (keeping track in the Word objects)
* Calculate a creativity ratio for the paragraph: (# unique words) / (total # words)
* The output lists the words and their counts then ends with the creativity ratio. (It may be easier for your testing to write this to a file.)

Extension (+5): Output the words in order by how often they appear (use a sorting algorithm, not brute force!)

* Extension points only possible if the program actually meets requirements