Hash Implementation Details

The hash table, implemented in Dictionary.java, was done as a table array of LinkedList buckets, with each entry containing a Word object implemented as a key-value pair. To handle collisions in the event that they occur, the buckets were implemented via separate chaining as a LinkedList where each Word in the LinkedList table points to the next in the event of collisions. Functions for getting, putting and removing were also implemented to allow for manipulation of the Dictionary as needed. The program is run from SpellCheck.java and its associated class, which handles the processing of the dictionary file, input text file, and output file and contains an instance of the Dictionary class for implementing these features. The process of actually checking the spelling of each text word and searching the dictionary for it, along with altering each word (such as adding/removing suffixes) to increase the likelihood of a successful dictionary is also handled within the SpellCheck class via a private checkSpelling method.

To allow for easier, more straightforward access to the generated keys, the hash and compression functions were implemented separately with the compression function being private and internal to the Dictionary hash table class. Keys are compressed as they are entered during operations and are passed in along with the compressed hash value and the String dictionary value. Due to the keys being strings of characters, the hash function was implemented as a 5-bit cyclic-shift, as this was shown to be an ideal implementation in *Data Structures and Algorithms in Java* for both ensuring excellent performance and minimizing collisions. Further, the compression function was implemented by applying the Multiply, Add and Divide (MAD) method to the aforementioned hashcode function by dividing by a set prime number with the purpose of arriving at a load factor (λ) of approximately 0.75. Another Dictionary constructor was also provided that allows future client code to specify another prime factor to be used to calculate a reasonable capacity, m, based on the size of the dictionary, n.