NAME: HAROLD DANE C. BAGUINON

DATE: 11/22/2014

DATE DUE: 11/24/2014 12:00:00 PM

COURSE: CSC136 010

PROFESSOR: DR. SPIEGEL

PROJECT: #4

FILENAME: P4 Design Documents.docx

PURPOSE: Design documents for orderedInsert() LinkedList function and p4.cpp’s function for removal  
 of a word.

1. orderedInsert() updates and functionality.
   1. Function must be updated to return a pointer to eltType.
   2. Check if LinkedList is empty or if value being passed is less than the data in the first node.
      1. If true, then a new node of value’s type is created, and it points to current first node, becoming the new first node when head is assigned to it.
      2. If neither is true, continue to else if.
   3. Compare value to first item in the LinkedList. If same, return pointer to data.
      1. If false, continue to else.
   4. New pointers are created called p and trailp, which are used to check nodes.
      1. p is set to point at second node.
      2. trailp is set to point at first node.
         1. Check if p is not pointing to NULL, and that the value passed is greater than the data in the second node.
            1. While this is true, move trailp up to p, and move p to the next node.
         2. Once p reaches NULL or the passed value is not greater than the data in the node, the while loop is exited.
         3. If p is still not NULL and if the passed value is equal to the value in p’s node, the pointer to p’s node’s data is returned.
         4. However, if the passed value is not equal to p’s data, a new node must be created which points to p. trailp->next is then set to point to the new node. In essence, the new node is inserted between p and trailp.
2. Removal of a word in the application.
   1. Prompt the user for a word.
   2. Call find() function from LinkedList class to receive returned pointer of desired word.
      1. find() will check each node by creating a pointer p, and comparing the passed value (a word) to each node’s data (a WordRec).
      2. Check to see if a value is the same as the first item in the LinkedList and if so, return a pointer to that item.
         1. Check if p is not pointing to NULL, and that the value passed is greater than the data in the second node.
            1. While this is true, move p to the next node.
            2. If the data in p’s node is the same as the passed value, the pointer to the data is returned.
            3. Otherwise, NULL is returned.
      3. If the word is found, inform the user of the number of times the word occurred in the input file.
         1. Ask the user how many copies of the word to delete.
            1. If that value is between 0 and the number of times the word occurred, then decrement the count by the value inputted by the user.
            2. If the value is equal to the number of times the word occurred, then the word’s WordRec is removed entirely with the remove() function.

The function operates similarly to orderedInsert.

Pointer p is set to head, and trailp is set to NULL.

p->data is compared to the word, and while it doesn’t match the word, trailp is set to p, and p is set to the next node.

Once p-> data equals the passed word, the while loop is exited.

If p is pointing to head, then head is moved forward, and p is deleted.

If p is not pointing to head, then trailp->next is moved up to p->next, and p is deleted.

* + - * 1. If the value is greater than the number of times the word occurred, print out error message, then main menu.
    1. If the word is not found, inform the user and print menu again.