Project 4 CS 136 Kutztown

Purpose: Complete a program to store words and their multiplicities in a linked list
Points: Program: 20 ; Design Documents: 5
Description: As before, process the words in a file, keeping track of the number of occurrences of each word. Then, permit the user to list words or parts of words under several scenarios, including several new menu items.

For your final project, you will complete a program whose files are provided to you. This program has similar functionality to Project 2, but uses a linked list to store data and permits removal of copies or entire words. The linked list is similar to that in class, but you will need to update it to, among other things, be able to handle duplicate entries.

Before you begin coding, create design documents for the orderedInsert() LinkedList function update and the application's function for removal of a word. These will be written algorithms, in outline form, describing the steps to complete each of these tasks and are expected to be fine grained, low level descriptions. You will submit them, via turnin, in doc or pdf form.

The functions you will be writing are noted below and in the files for the project, included in the Project4 directory on acad (a working executable is provided as well). They are also on the web, and require work as follows:

WordRec class:
operator (): The () operator is to be overloaded to returns the first *n* characters in the word data member. You are to complete this. You must use the proper function for the *string* class. You may not write your own version.

LinkedList class:
orderedInsert( ): This function now only inserts new data, returning void. It must be updated to return a pointer to eltType. If there is equivalence between the parameter **elt** and a data member of a node in the list, using the == operator, it will return a pointer to that node's data, i.e. that data (NOT the node); this permits the program at the class or application level to act accordingly. Otherwise it will return NULL, signifying the data was inserted in the list. The header is already updated to match these specs.

find( ): This function now returns bool, Update it to return a pointer to eltType according to whether the parameter was found. If the eltType parameter is found in the list, a pointer to that eltType (not the node) is returned. Otherwise NULL is returned, signifying the search was unsuccessful. The search must terminate as soon as the item is found or there is no longer a possibility of finding it. The header is already updated to match these specs.

remove(): This function is supposed to have a prerequisite that the item will be found, but in several places it contains code that isn't congruent to the prerequisite. Update the function so that it matches the prerequisite in its entirety.

Application p4.cpp:

Remove a Word: Add a menu item to remove a word (the choice MUST be R as in the menu in the box), the case to recognize the user choosing it, and complete the function in the application to handle it. The function will prompt the user for a word and will use the LinkedList find() function to determine if it is in the list.

If the word is found, the find() function will have provided a pointer to the WordRec object from which the count can be obtained. It will inform the user how many copies of the word appeared in the input file and

ask the user to input how many copies to delete. If that value is between 0 and the total occurrences of the word in the file, the count of the word's WordRec will be decremented accordingly. If the value equals the total occurrences of the word in the file, the word's WordRec is removed entirely. If the word was not found, the user is informed and the main menu reappears. If the number of copies to delete is out of range, an error message will be printed, followed by the main menu.

Read a File**:** Add a stream extraction operator at the location indicated in p4.cpp. Stream insertion operators don't have to be associated with a class. This one is associated with a particular instance of the LinkedList class and is therefore not appropriate to be included in the LinkedList class itself, if its generic nature is to be preserved. While it could actually be included at the beginning of the tpp file, it is as appropriate to place it in the application.

```
Choice:
A)ll Words with Number of Appearances
P)rint Words That Appeared a Specified # Times
S)how first N Characters of All Words
F)ind a Word
R)emove a Word
Q)uit
```

Notes:

- All files for this project are located in the Projects/Project4 directory of the public area on acad and on the web.
- The generic nature of the LinkedList class MUST be maintained. DO NOT import WordRec, or any other class that gives it specificity towards the data it holds. Substantial penalties (at least two letter grades) for non-compliance.
- orderedInsert() will require restructuring, as inserting to an empty list is absolute, while other inserts are conditional on whether the item is found in the list.
- Before beginning the project, you should run the executable and familiarize yourself with all files and functions. Not only will this help you understand the operations, any part of this program may be included on the final exam. In particular, the list iterator has significant additions of functions that you should be able to explain.
- The design documents may be handwritten, if they are neat.
- Your program will be run using a script, so your menu MUST match the one in the box, and in particular must use the same letters (case insensitive) for the choices and the same prompts as in the demo executable. Penalty for non-compliance is two letter grades.
- You may use your own version of the WordRec class. This may save time and effort in documentation, but be sure all necessary operator overloads are completed and the documentation is properly updated for this project.
- Your program must be well written, properly indented, and commented, including an id block, in all files. This means that you are expected to update the documentation in every file and upgrade the id block while retaining proper attribution. A letter-grade penalty will be assessed for each file without a proper id block, complete with description.

Deliverables: Submit the following items, using the turnin utility

Design Documents: One doc/docx or pdf file, by noon on Saturday, November 22.

Program Files: All code files: Application **p4.cpp, WordRec** class and **LinkedList** class, along with **LinkedList.tpp**. The **makefile**. Do not turnin temp.cpp. 2-point penalty for non-compliance on exact file names, 1 more for not turning in the makefile. Due at noon on Monday, Dec. 1

Bonus: 3 point bonus for submission of design documents before 12 Noon on Saturday, November 20.
6 point bonus for submitting updated LinkedList class with test application using LinkedList<int> that can be tested via input from the user by 8 PM on Sunday, November 23.