CS 315 Data Structures **Spring 2008**

Project # 4 - Final version

Due: April 22, 2008

Problem Statement:

You are to implement a spell checker that takes as input a dictionary D(a list of words in English) and a text file T, and output the words in T that do not appear in D. For each word that does not appear, your program should also suggest possible correct spellings.

Goals of the project:

- Learn to implement or use a hash table class.
- Learn some basic strings operations.
- Learn to compare two different programming techniques experimentally.

Implementation Details:

Your program should insert the words in the dictionary D into a hash table using (a) closed hashing and (b) chaining. As a first step, count the number of words in the dictionary and use this information to create the hash table size so that the load factor ~ 1 in the case of chaining and ~ 0.1 in the case of closed hashing. Insert all the words in T into the hash table. Then search for each word w in T in the hash table. *The output is a list of words w in T, but not in D.*

For each such word, the following approach is used to guess the correct spelling: for each word misspelled word w, let W be the list of words that can be obtained by changing, inserting or deleting one letter from w, by interchanging two adjacent letters of w or by replacing one of the upper-case letters appearing (at a position other than the first). Next, each word x in W is searched in the hash table and all the searched words that appear in T are suggested as possible choices for the correct spellling of w.

You need not implement a hash table class. You can use a hash table implementation that is publicly available, for example, from Mark Weiss's web site *(and discussed in the text). STL seems to provide an implementation of chaining based (open) hashing, but not closed hashing so it is better to use the hashtable class provided in the text. The code provided by Weiss includes closed hashing, but it uses quadratic probing for collision resolution. You are to modify it to implement linear probing.*

In the case of closed hashing, you are to use linear probing with an increment d > 1. You can determine d experimentally.

Some of the supporting functions that need to be implemented are as follows:

- Remove punctuations: if a word ends with ? , ! etc., these characters should be removed before the word is searched. *While reading words from the input file, blank space will be used as a delimiter to separate adjacent words. Thus, you can assume that a punctuation is also required to be followed by a space.*

- Change the upper case to lower case: If a word contains an upper-case letter (for example, in the beginning of a sentence) it should be changed to lower-case before searching.

Your program should keep track of the time taken to perform the spell-check using two different implementations of hashing – closed hashing and chaining. In each case, the time taken should include the insertion of the words in D into the hash table as well as the time taken to search all the words of the text and the alternatives to the misspelled words.

Input/Output format:

Your program should take the file names (for D and T) on the command line and should display the misspelled words. For each misspelled word, it should suggest possible correct words. The following sample input/output should be used as a specification of the input/output format for your program.

Sample input and output:

Assume that input.txt contains

```
Hlllo there mate! How aer you diong toDay? That's great mna. Good for yoo.

% spelling dict.txt input.txt

output of closed hashing:

Misspelled words (with suggested corrections)
----------------
Hlllo --> hello
aer   --> air, are, her, per
diong --> doing
toDay --> today
mna   --> man
yoo   --> you

Num Misspelled: 6
Time: 6.10001e-05 seconds

output of chaining :

Misspelled words (with suggested corrections)
----------------
Hlllo --> hello
aer   --> air, are, her, per
diong --> doing
toDay --> today
mna   --> man
yoo   --> you

Num Misspelled: 6
Time: 4.30001e-05 seconds
```