

---

---

**CSCE 210/2201 Fall 2018 Dr. Amr Goneid**  
**Assignment #4**

---

---

*Date: Mon Oct 22, Due: Thu Nov 1, 2018*

---

---

**The problem: Spell checker, an application of hashing**

Many applications such as word processors, text editors, and email clients, include a spell-checking feature. A spell checker's task is centered on searching for words in a potentially large wordlist and reports all of the misspelled words (i.e. words that are not found in the wordlist). An efficient search is critical to the performance of the spell checker, since it will be searching the wordlist not only for the words in the input, but possibly suggestions that it might like to make for each misspelling.



**Objective**

Implement a spell checker that reads a wordlist from a file ([wordlist.txt](#)), stores the list of words in a hash table using separate chaining (i.e. linked lists) to resolve collisions, then spell-check an input file. For each misspelling found in the input file, your program will report it, along with a list of 5 suggestions (i.e. 5 closest words from the wordlist).

**Suggested closest words**

Definition of closest in this assignment follows the *Hamming-Distance* concept. Distance between two strings follows the algorithm below:

*For two strings A, B.*

*A match occurs at position k if  $A(k) = B(k)$ .*

*$M = \#$  of matches*

*Distance = Number of mismatches  $D = \max(\text{length}(A), \text{length}(B)) - M$*

*$D = 0$  if A and B are identical*

For each misspelled word, starting with the same first character find the list of **five** closest words from the wordlist using the above algorithm.

**Hashing function**

Your spell checker is **case insensitive**. The hashing function is  $h_2(h_1(\text{key}))$  where key is the string such that:

- The **Hash code map**:  $h_1(S) = \sum_{i=0}^{L-1} \text{int}(S_i) \alpha^i$ , where  $\alpha = 26$ .
- The **Compression Map**:  $h_2(K) \rightarrow [0, N-1]$ , where  $N$  is the table size

**The hashTable ADT**

The *hashTable* ADT can be implemented using a Simple Linked List (*SLL*) with the following member functions:

- **Constructor:** Construct an empty table
- **Destructor:** Destroy table
- **MakeTableEmpty:** Empty whole table
- **TableIsEmpty :** Return True if table is empty
- **TableIsFull :** Return True if table is full
- **Occupancy:** Return number of occupied slots
- **Insert:** Insert key and data in a slot
- **Search:** Search for a key
- **Retrieve:** Retrieve the data part of the current slot
- **Update:** Update the data part of the current slot
- **Traverse:** Traverse whole table

### **Required Implementation:**

1. Design and implement a template class *hashTable* using a SLL with a minimum of the above member functions.
2. Read and store each entry in the file [wordlist.txt](#) in the hash table using separate chaining to resolve collisions.
3. Spell check an input file such that for each misspelled word list 5 suggestions from the wordlist that are closest to it based on the Hamming Distance measure.