Course: CS 3460: Data Structures (and Algorithms) Spring 2017
Assignment 03: Fun with Hash Tables
Due Date: Wednesday, September 27; 23:00 pm

Spellchecker (10 points): We will be building a rudimentary spell checker, one that is capable of suggesting alternate spellings to words that are mis-spelled by exactly one character. This is achieved by first building a string set data structure implemented as a hash table, that is capable of storing all words in the English dictionary. It will then be possible to query this data structure for all possible alternate spellings of a word.

Getting Started: Please download the files `StringNode.java`, `StringSet.java`, `SpellChecker.java` and `dictionary` into your working directory. The `StringNode` class provides the basic node structure for this assignment. It is already implemented with a basic constructor and appropriate get and set methods. You should be able to work this assignment without modifying this class. But feel free to modify it with additional methods if you think it will be beneficial for your implementation of the spell checker.

The StringSet class: Complete the `StringSet` class, so that it fully supports operations `insert`, `find` and `print`. You also need to complete the method `hash` that hashes an input String to valid table indices. For this, you need to use a polynomial hash function as discussed in class. You may evaluate this polynomial at any random prime number in the range $0 \ldots 10000$. For a list of prime numbers, see this Wikipedia page.

Note that the `StringSet` class contains a constructor that only allocates a table of size 100. If the number of elements stored in this table reaches its capacity, then we will expand the table to twice its original size, and re-hash all the elements. The code for expanding the table size should be written in the `insert` function, at the appropriate check:

```
if (size == capacity) {
  // code for expanding the table.
  ...
}
```

You may try to implement the above expansion code after you have tested all the other methods (including `insert`) in the `StringSet` class. The class should still be functional without the expansion code, but will be less efficient.

SpellChecker Class: This class contains the `main` method of the spell checker application. A lot of code is already provided to get you started on the application. Specifically, an object of your `StringSet` class is declared, and is loaded with words from the dictionary. If the table expansion code is written correctly, this should take less than a second to execute.

The main while loop waits for user input. Please finish up the code that provides alternate spellings to the user input that differ in at most one character. Note that only words that can be obtained by modifying at most one character are suggested, not words that are obtained by adding or removing characters. An example output is shown below.

```
Dicitonary loaded...
```

```
algorithm
algorithm is correct.
algorithx
Suggesting alternatives ...
algorithm
coee
Suggesting alternatives ...
code
coke
come
cone
cope
core
cote
cove
coed
```

Note: Pressing `Ctrl-d` marks the end of input and terminates the main while loop. You may also find the `StringBuffer` class in the Java library useful, particularly its `setCharAt` method.

Submission: Please submit a single compressed file named `hw04.zip` that contains `StringNode.java`, `StringSet.java` and `SpellChecker.java` through ASULEarn.

Input/ Output Instructions: For all programs, until and otherwise stated, we will be taking the input from standard input (`System.in`) and will be sending the output to standard output (`System.out`).

Notes on Coding: Please do not include user-defined packages in your code. Your code should run in the Unix/Linux machine using the commands `javac` and `java`.