# Project 3 README – The Spell Checker

The Spell Checker program will check a text file of words and when it finds a word that is not in the dictionary, the program will allow the user to take one of several actions. The actions are, as follows:

* Add the word to the dictionary
* Ignore the word and not add it to the dictionary
* Go on to the next word
* Search for a suggested spelling in the dictionary
* Quit spell checking the file

When the Spell Checker program is closed, it will output all words not found in the dictionary and the line number on which they occurred on to an external file. The dictionary for the program will also be updated with any new words that the user added.

## Design

The program uses two input files, one for the dictionary and another for the text file that is being spell checked. The program will produce one output file and update one of the input files. The output file will be a text file of all words, sorted alphabetically, not found in the dictionary and the line each word was found one. The dictionary file will be updated to include new words that the user added during program execution.

The program utilizes a BinarySearchTree class to implement the sorted list of words that were not found in the dictionary, a HashTable class, which is used within the Dictionary class. The Dictionary class is used to store the words found in the dictionary file, if the file contains any words. The LinkList class is also utilized in the implementation of the HashTable class.

The main class this program focused on is the Dictionary class. The class itself only included 4 public methods, one of which was a constructor with initialization of the not found value and the size, both values used to create the HashTable. The HashTable object for the Dictionary class was a private variable, so the client file could not access the HashTable object directly but had to do it through the other 3 public functions.

The 3 public methods were insert, lookup and makeEmpty. The insert method took a HashedObjext as it’s only parameter and this was the key, in terms of the HashTable. The lookup method is similar to the insert method, but in reverse. The lookup method would lookup and return a HashObject when passed the key for that HashedObject. The makeEmpty took no parameters and could be used to empty the dictionary file.

The SpellChecker program considers a word to be any set of characters or symbols that are surrounded by whitespaces. This criterion forces the program to grab words, even if symbols are within the word, then be able to filter out the symbols, so only the word itself is left.

The program will check every word to ensure it’s in the program’s dictionary. If the word is not in the dictionary, then the user is asked what to do. The program can suggest an alternate spelling for the word, add the word to the dictionary or ignore the word.

## Algorithm

1. Program is started by the user and file to be spell checked is selected.
   1. Start up includes loading the dictionary and all other external files.
   2. If no external dictionary file is available, then the program dictionary will start out empty.
2. The program will read the external file word by word, until it finds a word that is not in it’s dictionary.
3. Store the word not found in the program dictionary separately from the dictionary.
4. The program will prompt the user for an action for the word that was not in the program dictionary. After the action is executed, the program should continue spell checking the file, unless the user selected to quit the program.
   1. If user selects A, then add the word to the program’s dictionary.
      1. The word should be added to a separate dictionary of words that need to be added to the program dictionary.
   2. If user selects I, then ignore the word and all other occurrences of that word in the file.
   3. If user selects G, go on to the next word.
   4. If user selects S, search for a suggested spelling.
      1. Adjacent letters in the word should be swapped, either until swapping the letters creates a word found in the dictionary or until all possible swaps have been executed.
      2. If swapping letters creates a word that is in the program dictionary, display the word with the swapped letters to the user as a suggested spelling.
      3. If swapping letters does not create a word found in the program dictionary, alert the user that no suggested spelling can be found for the word and continue to the program.
   5. If user selects Q, start the end of the program.
5. All words that the user wants added to the program dictionary, should be appended to the external dictionary file.
6. All words that were not found in the program dictionary, should be stored in alphabetical order by word in an external file. The word and the line number the word was found on should be stored.
   1. If a word appeared multiple times in the file to be spelled checked, then only the last line number occurrence of the word should be stored in this file.
7. Close Program.