The Problem:

The problem for this assignment is to create a spell checker using binary search trees to create a dictionary. After the dictionary is created, we will parse in a document to be checked word by word.

The Algorithms:

First, we will need to create a few global variables labelled found, notFound, foundComparisons, and notFoundComparisons. These will be used to count their respective values. We will also create an array of Binary Search Trees labelled dictionary.

Second, we will create a method to make the dictionary named createDictionary that will take an argument of a String which will be the file name. In this, we will set up a loop to create 26 Binary Search Trees. Then we will read the file one word at a time, entering each word into its respective binary search tree based on the first letter of the word.

Third, we will create another method named parser that takes an argument of a String which will be the file name we will check for spelling errors. This will scan the document letter by letter until a white space is reached. When this happens, it will find the binary tree the word should be in, and traverse the tree to see if it finds the word. If it finds the word, the found and foundComparisons get added to. If the word is not found, the notFound and notFound Comparisons get added to. Then it continues testing each word until it reaches the end of the document.

Finally, we will create a class called SPClient to hold the main and run the program. This will allow us to run it from the command line and input args for the file names.

Results:

```
C:\Users\rwohl\Documents\NetBeansProjects\Lab7\src>java SPClient random_dictionary.txt oliver.txt
Words found: 937492
Words not found: 54648
Comparisons for words found: 14388139
Comparisons for words not found: 568211
Average Comparisons per word found: 15.35
Average Comparisons per word not found: 10.40
Total time: 7.322 seconds
```

Observations:

I noticed that the Binary Search Tree method is much quicker than using Linked List. After thinking about it, it makes sense because the binary search tree will narrow down the search each time it continues down another node while the linked list checks every node possible until it finds a match. This cuts the time by about a minute. This exercise shows that each type of container is useful in their own ways, while other might be better in different implementations.