University of Technology, Sydney

Faculty of Engineering and Information Technology
School of Software

31251 – DATA STRUCTURES AND ALGORITHMS

AUTUMN 20015
ASSIGNMENT 2 SPECIFICATION

DUE DATE – 20th May 2015
FINAL DUE DATE – 26th May 2015

This assignment is worth 35% of the total marks for this subject.

ASSIGNMENT DOCUMNTS

This assignment is split into two documents.

1. *Assignment Specification*. This document contains the specification for what you will be required to do.
2. *Assignment Addendum*. This document contains all other details regarding the assignment, including assignment submission and the assessment scheme.

REQUIREMENT

For this assignment you are to write a program that reads a text file, identifies the words in it and then checks the spelling of each word against a small dictionary of words. Any words not found the dictionary are printed out. Finally, the total number of words in the file is printed out and the total number with unknown spelling.

DEFINITIONS

Text Files

For our purposes, a text file consists of lines of text containing words, spaces and punctuation with a newline character at the end of each line. All characters will be from the basic ASCII character set.

Words

- Words are sequences of alphabetic and numeric characters, apostrophe, the underscore and hyphen characters that are separated by sequences of one or more separation characters. See below for a list of the separator characters.
- The input for this assignment will consist of normal words plus integer and floating point numbers.

Separator Characters

- The list of word separator characters includes the following :
  space, tab (`\t`), newline (`\n`), carriage return (`\r`)
  and the common punctuation characters
  `,;:"~!#%^*()=+[]{}\|<>?/`
- If the `.` character has a space, tab, newline, digit, minus or plus on the left and a digit on the right then it is treated as a decimal point and thus part of a number. Otherwise it is treated as a full stop and a word separator.

IMPLEMENTATION DETAILS

1. Assuming your compiled program is called `a.out` your program will use command line argument as follows
   ```
   ./a.out dictionaryfile textfile
   ```
   Where `dictionaryfile` is a file containing a list of correctly spelled words and `textfile` is the file containing the text you are going to spell check.

2. In the SUBNET assign directory you will find a file called `dict.txt`. This file contains the 1000 most commonly used words in the English language, all in lower case and with one word per line. You will use this as your dictionary file. Do NOT change the contents of it.

3. Your program should start by reading the dictionary file, loading each word into a C++ string and then storing that string in a binary tree. A copy of the `bintree.h` and `binnode.h` files can be found in the SUBNET assign directory.
   PLEASE NOTE 1. You must use these binary tree classes and you may not alter them in any way. Do NOT use any other data structures to store the word list.
   PLEASE NOTE 2. To use them you will just `#include bintree.h` in your code. Do NOT cut and paste the entire bintree code into your code.

4. Your program will then read the text file, line by line, and extract every word from the line. Each extracted word will be checked to see if it exists in the tree. If not then the word will be printed. Please note the following regarding extracted words
   - Before checking a word against the tree, all letters in it will be changed to lower case. That is "`Hello`" will be changed to "`hello`" before seeing if it exists in the tree.
   - If any extracted word is a number, either an integer or real, it will be ignored. That is, it won't be checked for spelling. Look up the `strtod` library function to work out how to do this.

5. Finally, your program will print the total number of words in the file and then print the total number of words with unknown spelling. Numbers are not considered to be words and thus not included in the count of words.

BENCHMARK PROGRAM

To clarify how the program works, a benchmark executable version has been placed on the server. You can run it by typing the following command

```
/home/glingard/spellcheck
```

You will need to give it a dictionaryfile and textfile to check for spelling. To give you an idea of the size of this assignment, the source code for the benchmark program, including whitespace and comments, but not including the bintree code, takes approximately 280 lines of code.

ASSIGNMENT OBJECTIVES

The purpose of this assignment is to demonstrate competence in the following skills.

- Program design – including class design
- Using command line parameters
- String manipulation
- File handling

- Tree manipulation

These tasks reflect all the subject objectives apart from objective 4.

As part of your subject workload assessment, it is estimated this assignment will take 35 hours to complete.