COSC 2320: Data Structures

Homework 4: Sorting and Searching

Deadline: **Nov. 8, 2014**

1. INTRODUCTION

In this homework you are asked to create a C++ program that implements a primitive spell-checker, using efficient sorting and searching algorithms.

In essence a spell checker works by comparing all the words in a document to words in a "dictionary". If a word is not found in the dictionary, the program assumes that it is a misspelling. In order to efficiently spell check a document you need to have your dictionary in a **sorted array**, and you need an efficient way of determining if a word can be found or not in the dictionary array.

2. INPUT & OUTPUT

The input of this program is two files, whose names are given in the command-line arguments. The first file provided will be the dictionary file, and the second file will be the file in which you will check the spelling. Both files come with a list of words separated by spaces or carriage returns and they both can have arbitrary number of words. Characters other than letters and numbers will not be added into the files.

You will first read the dictionary file into a linked list. Once the file is read and the number of words is known, you should allocate an array and use it to sort your dictionary file if it is not already sorted (this could be verified while you move it to the array). If there are duplicate words, only one instance of the word should be kept. You should also delete the original list to free memory.

Then the file to be checked will be loaded into memory in the same way as in previous homework. You should use an efficient search algorithm (Binary Search or AVL Binary Search Trees) to determine if each word of the input file is present in the dictionary. You should produce a list in **ALPHABETICAL ORDER** with all the words that are misspelled in the document, one word per line. If there are misspelled words appeared in the file more than once, only one instance will be shown in the misspelled list. The spell checker is **case-insensitive**.

3. PROGRAM AND ARGUMENT SPECIFICATION

The main program should be called "**spellchecker**". The program should be able to take the first argument as the dictionary file name and the second argument as the name of the file to be checked.

The call syntax will be like:

```
spellchecker.exe dict.txt doc.txt
```

Note that the file names will not necessarily be the same every time, so your program shouldn't have those file names hard coded.

All the words in the dictionary file will be in lower case.

4. A SAMPLE TEST CASE

dict.txt:

```
apple air zoo kite hour picture
friend city
game my
table dinner party
```

input.txt:

```
Apples hour picture frined table
Dinner
```

Program call:

```
spellchecker.exe dict.txt input.txt
```

Output:

```
Apples
frined
```

5. SUBMISSION REQUIREMENTS

Your submission should be well tested before submitting under Visual Studio 2010 or later versions. You can get a copy of Visual Studio from the UH website using your cougarnet username and password. The URL is http://uh.edu/infotech/php/software/list.php.

We use the UH blackboard system to collect your homework submissions. Before you submit your homework, please make sure to **put everything in a ZIP file** named in the form of **LastName_PeopleSoftID_HW4.zip.**

For example: Zhang_1234567_HW4.zip

The instructions about how to use the blackboard system can be found on the TA's webpage for this course: http://www2.cs.uh.edu/~yzhang/cosc2320-f2014/

6. GRADING

The maximum grade for this homework is 100.

You will get 15 pts for submitting the homework in time, 10 pts if your program can be successfully compiled.

We will test your program with 5 easy test cases and 5 hard test cases. Each easy test case will worth 10 pts, and each of the hard ones will worth 5 pts.

When testing, we will compare your program's output with the standard output. Therefore **do not print any content on the screen unless required**, avoid any prompt information like "Please enter the input file name:", "The elements in the doubly linked list are:" etc.

Last but not least, **no cheating or plagiarism will be tolerated in any graded submissions.**