**CSCI 3412 – Algorithms**
**Dr. Williams**
**Program 2 – Searching Algorithms**
**Due March 9, 2015**

In Problem Set 2, each student will analyze and implement four (4) different search algorithms. There will be static analysis of the algorithms – based on the written pseudo-code – as well as analysis of the performance of the algorithms – based on the execution of the code.

I will give you an input file, wordlist.txt, that contains the complete works of Shakespeare with a single word or punctuation per line as well as blank lines where they appear in the text. A string is considered to be a word if the first character is alphabetic, although characters like hyphens and quotes may occur within a word. [This isn't perfect, but it is simple to implement and will ensure everyone uses the same rules and gets the same counts.] Convert each word to lowercase to ensure that differences in case don't lead to different words. It is only the words we are interested in – that is, don't add punctuation to the concordance.

Here is a sample from the beginning and end of the file.

```
A
MIDSUMMER-NIGHT'S
DREAM

Now
,
fair
Hippolyta
,
our

…

state
,
That
like
events
may
ne'er
it
ruinate
.
```

Each student will create a concordance of the complete works of Shakespeare using the given word list. A concordance is a sorted list of the words with a count of the number of occurrences of each word. For this program, you will also need to print the number of unique words found, the first ten (10) words, and the last ten (10) words, to verify that the algorithm is correct.

For this data set and using the rules above, the first ten (10) and last ten (10) words alphabetically and their counts are:

Alphabetic
Number of unique words = 27886
"a" : 13679
"a'" : 120
"a's" : 1
"a-bat-fowling" : 1
"a-bed" : 12
"a-birding" : 4
"a-bleeding" : 2
"a-breeding" : 1
"a-brewing" : 1
"a-broach" : 1
…
"zenelophon" : 1
"zenith" : 1
"zephyrs" : 1
"zo" : 1
"zodiac" : 1
"zodiacs" . 1
"zone" : 1
"zounds" : 1
"zur" : 2
"zwaggered" : 1

**The Search Algorithms**

Each student will select and implement search algorithms to create the concordance based on the following data structures.

1. Unsorted sequence (vector or list)
2. Sorted sequence (vector or list)
3. Binary search tree
4. Hash table

For the first two – unsorted and sorted sequence, you may use either a vector or list based implementation. Also, the sorted sequence and binary search tree naturally maintain the concordance in sorted order. For the other two, you will have to sort the concordance following its construction.

Note that it is the searching algorithm used in building the concordance whose analysis we are interested in – not the sorting phase (for the unsorted and hash table algorithms).

**Part 1 – Write Pseudo-Code for each of the algorithms**

Write (in the same style as the text) the pseudo-code for the search algorithms. Most – if not all – of these are in the text or available from other sources. Properly document the source of your algorithms. Also, give an estimate of the asymptotic growth of each algorithm based on an informal analysis of the pseudo-code.

Note that it is only the search algorithms you need to document.

**Part 2 – Implementation**

Implement each of the algorithms in a programming language of your choice. In order to perform the analysis in Part 3, you will need to instrument the algorithms to count the number of comparisons performed and the number of array (or list) assignments.

**Part 3 – Analysis**

Using the given word list, analyze how your four algorithms perform. Specifically, show the number of comparisons and assignments used in constructing the concordance. How does that compare with your original expectation?

Specifics

Each student will submit a report for the project as a single pdf file. The report will contain:

1. Problem Description
2. Algorithm Design
3. Implementation – including source code
4. Analysis
5. Conclusions