

This assignment has several parts -- a comparison of dictionary/map ADTs (section 5.2.3.3 of the text) and a concordance-production application using the dictionary ADTs. A Webster's dictionary definition of concordance is: "an alphabetical list of the main words in a work." In addition to the main words, I want you to keep track of all the line numbers where these main words occur.

WORD & LINE CONCORDANCE APPLICATION

The goal of this assignment is to process a textual, data file (`hw4data.txt`) to generate a word concordance with line numbers for each main word. A dictionary ADT is perfect to store the word concordance with the word being the dictionary *key* and a list of its line numbers being the associated *value* with the key. Since the concordance should only keep track of the "main" words, there will actually be a second stop-words file (`stop_words.txt`). The stop-words file will contain a list of stop words (e.g., "a", "the", etc.) -- these words will **not** be included in the concordance even if they do appear in the data file. Sample files might be:

Sample <code>stop_words.txt</code> file	Sample <code>hw4data.txt</code> file	Sample output file
<pre>a about be by can do i in is it of on the this to was</pre>	<pre>This is a sample data (text) file to be processed by your word-concordance program. The real data file is much bigger.</pre>	<pre>bigger: 4 concordance: 2 data: 1 4 file: 1 4 much: 4 processed: 2 program: 2 real: 4 sample: 1 text: 1 word: 2 your: 2</pre>
<p><u>Notes:</u></p> <ol style="list-style-type: none"> 1) Words are defined to be sequences of letters delimited by any non-letter. (e.g., white space, punctuation, parentheses, dashes, double quotes, etc.) 2) There is to be no distinction made between upper and lower case letters. (e.g., "CAT" is the same word as "cat") 3) Blank lines are to be counted in the line numbering. (e.g., line 3 above is blank) 		

The general algorithm for the word-concordance program is:

- 1) Read the `stop_words.txt` file into a dictionary containing only stop words, called `stopWordDict`. (WARNING: Strip the newline ('`\n`') character from the end of the stop word before adding it to `stopWordDict`)
- 2) Process the `hw4data.txt` file one line at a time to build the word-concordance dictionary (called `wordConcordanceDict`) containing "main" words for the keys with a list of their associated line numbers as their values. The main loop is something like:

```
lineCounter = 1
for each line in the data file do
    processLine( lineCounter, line, wordConcordanceDict... )
    lineCounter += 1
```
- 3) Traverse the `wordConcordanceDict` alphabetically by key to generate a text file containing the concordance words printed out in alphabetical order along with their corresponding line numbers.

The general algorithm for the `processLine (lineCounter, line, wordConcordanceDict ...)` function is:

```
wordList = createWordList(line)
for each word in the wordList do
    if the word is not in the stopWordDict then
        if the word is in the wordConcordanceDict then
            look up the line-#-list value associated with the word in the wordConcordanceDict
            append the lineCounter to the end of the line-#-list
        else
            add the word with an associated [lineCounter] list value (e.g., list containing first line #) to the
            wordConcordanceDict
```

(Note: I strongly suggested that the logic for reading words and assigning line numbers to them be developed and tested separately from other aspects of the program. This could be accomplished by reading a sample file and printing out the words recognized with their corresponding line numbers without any other word processing.)

DICTIONARY ADT COMPARISON

We have 3 dictionary ADT implementations from lab 7: `ListDict`, `ChainingDict`, and `OpenAddrHashDict`.

None of these should need to be modified. You just use their dictionary operations.

Time your word-concordance application using all three dictionary ADT implementations to complete the following table: (FYI, there are about 2,700 stop words and less than 200 non-stop words)

Dictionary ADT Implementation Used	Word-concordance Program Execution Time (seconds)
<code>ListDict</code>	
<code>ChainingDict</code> (hash table sizes 4096)	
<code>OpenAddrHashDict</code> with linear probing (hash table sizes 4096)	
<code>OpenAddrHashDict</code> with quadratic probing (hash table sizes 4096)	

DATA FILES - Download hw4.zip file at <http://www.cs.uni.edu/~fienup/cs1520s13/homework/> it contains:

- `ListDict` in the file `list_dictionary.py`, `ChainingDict` in the file `chaining_dictionary.py` and `OpenAddrHashDict` in the file `open_addr_hash_dictionary.py`
- the stop words in the file `stop_words.txt`
- the data file to be processed by your word-concordance program in the file `hw4data.txt`

EXTRA CREDIT POSSIBILITIES:

- 1) Use a better definition of a word that allows words to contain an apostrophe or single hyphens. For example, "it's" and "end-of-line-characters" should be considered words.
- 2) Modify `OpenAddrHashDict` dictionary ADT to allow double hashing as a rehashing technique.
- 3) Modify `OpenAddrHashDict` dictionary ADT to allow the capacity of hash table to double if the hash table load factor exceeds 0.8.

SUBMISSION

Submit **ALL necessary files** to run your concordance-production application using the dictionary ADTs as a single zipped file (called hw4.zip) electronically at

https://www.cs.uni.edu/~schafer/submit/which_course.cgi

Include in your hw4.zip file a "results" file (hw4.doc, .txt, .rtf, .odt, etc.) containing the completed table above, i.e., timing results for your word-concordance programming using the various dictionary ADTs.