Due date:

This file contains Lab 4. You must submit your answers to the D2L Dropbox "Lab-4" by the end of today's lab.

Lab 4 requires Java programming. You can work in pairs (but you must still submit your own work to D2L).

Note that late assignments will not be graded.

Please do not zip or compress your submissions. D2L allows you to upload multiple files

Grading:

- Applying sequential search for solving problem [3 mark]
- Applying binary search for solving problem [4 mark]
- Input/ output and file processing [2 mark]
- Short report [1 mark]

You need to hand in the following to D2L:

1. A short report outlining your results, and conclusion.
2. Your java codes.
3. Print screen of your outputs.

In this lab, you will write a spellchecker. Actually, you will write two spellcheckers:

- One using brute force (sequential search)
- One using decrease and conquer (binary search)

You must proceed as follows:

1. First, get these files from D2L:
   - "lab4_wordlist.txt"
   - "lab4_testdata.txt"
2. Create a file *SpellChecker.java* with three methods*: main(), seqSearch(), binSearch()*.
3. Your program should take two text files as input ("lab4_wordlist.txt" and "lab4_testdata.txt")
4. SeqSearch() method should use sequential search algorithm and try to find each word in "lab4_testdata.txt" in the "lab4_wordlist.txt" file. Keep track of the number of words you can't find. (Ignore Capitalization.)
5. BinSearch() should use binary search and try to find each word in "lab4_testdata.txt" in the "lab4_wordlist.txt" file. Keep track of the number of words you can't find. (Ignore Capitalization.)
6. For each search, determine how long it takes to check all words. Make sure you time only the spell checking, not the input/output or file processing.

Input to your program: the words in the two files posted on D2L.
Output from your program: for each search algorithm you test, you must output (to the console)

a. the number of word(s) you couldn't find
b. the time (in Microsecond) that it took to find (a)

For example, I might output (from one of two algorithms): Note: this is not the correct answer!

Binary search: 10 words 800.789 Microsecond

You should conclude:
Which algorithm is faster (binary search or sequential search)?
What is the overall efficiency class for each algorithm? (Like O (n^2), O (nlogn)...)