CSC2001F

Data Structures 1: Assignment 1

Mutual Friend Search

Duration: 7 days

[Start: 3 March 2014; **Due:** 10 March 2014 9am]

Suppose a very simple social network stores friend lists as text files for each user. You are provided with two text files*, toSearchFor.txt* and *toSearchIn.txt*. *toSearchFor.txt* contains an unsorted list of Alice's friends; *toSearchIn.txt* contains a list of Bob's friends, sorted in alphabetical order. You are to determine who Alice and Bob's mutual friends are by searching *toSearchIn.txt* for the friends listed in *toSearchFor.txt*. This should be done according to the task descriptions below.

Task 1:

Write a **sequential search** method which takes in the name of each friend in *toSearchFor.txt* and searches for it in *toSearchIn.txt*. If the name is in *toSearchIn.txt*, the person is a mutual friend and the method returns the position of this name (although your array of names will begin at zero, the position of the name in slot zero, is 1). If the name is not found, the method returns -1. Also keep track of the number of comparisons performed for each name. Since the names are sorted, if while looking for name X you reach a name Y such that Y > X, you need to stop looking because obviously X is not in the file. Also show the time taken to complete the search for each name. **Hint:** use System.nanoTime() and then convert to milliseconds and print out the time. Format to 2 decimal places.

Output:
Print the name, position, number of comparisons and time, for each name in *toSearchFor.txt*. Also display the total time taken to find all mutual friends and the total number of comparisons. An output sample is provided below. Do not worry about formatting the output into perfectly aligned columns. Note that your value for time may vary slightly based on your computer.

```
Sequential search
Name            Position        Comparisons     Time (ms)
Cheryl Green    165             165             0.04
Karen Bell       -1             530             0.09
David Lee       211             211             0.03
.
.
Total Comparisons to find mutual friends: 47540
Total Time to find mutual friends: 5.78
```

Task 2:

Write a **binary search** method which takes each name in *toSearchFor.txt* and searches for it in *toSearchIn.txt*. If the person is a mutual friend, the method returns the position of this name in *toSearchIn.txt*. If it is not found, the method returns -1. Again, keep track of the number of comparisons for each name. In implementing the binary search, use the standard three-way comparison approach, i.e. the first version in the book, not the improved second version.

Output:

Output the results in the same format as in Task 1, but round the time to 3 decimal places instead of 2. See the sample output below:

```
Binary Search
Name            Position        Comparisons     Time (ms)
Cheryl Green    165             9               0.026
Karen Bell      -1              10              0.01
David Lee       211             20              0.011
.
.
Total Comparisons to Find Mutual Friends: 935
Total time to Find Mutual Friends: 0.613
```