Indiana University
CSCI-C212: Introduction to Software Systems (S25) Lab 13
Joshua Crotts and Muazzam Siddiqui

Searching/Sorting

What To Do:

Follow each step carefully. After finishing, please submit your work, as a ZIP, to Canvas, and let one of the AIs know.

Questions

1. Recall the sorting algorithms from lecture. In the starter code for today's lab is a dictionary of almost 500,000 words. The problem is that this dictionary was destroyed and shuffled by a hacker. Your job is to implement the generic static `mySort` method, which returns a list of the sorted strings.

   In this lab you will primarily work with Java's `main` method. There are no associated JUnit tests.

   (a) First, read the file into a list and sort it using `mySort`. You can use any of the five sorting algorithms from the textbook/lecture, or another that you know of. Do **not** modify the input list. (Therefore, you should use a *functional* sorting algorithm.)

   Warning: remember the analysis of the sorting algorithms from section 7.1. Insertion and selection sort grow, in the worst-case, quadratically. Merge and quick sort do not and are much faster. If your code isn't sorting the lists in under a second or so, then you need to use a different algorithm!

   Another Warning: do **not** use code from any other source aside from the textbook. You cannot use ChatGPT/Google/any other means. Read section 6.2. If you do, you will receive a 0 on this lab.

   (b) Second, sort the list using the built-in `Collections.sort` method. This method *will* modify the input list.

   (c) Output (using `System.out.println`) the time difference, in milliseconds, between your method's sorting approach and Java's. How big of a difference is there?

   Hint: to compute the time elapsed before and after a method call, use the Java-provided method `System.currentTimeMillis()`.

   ```
   long before = System.currentTimeMillis();
   callToSomeMethod();
   long elapsed = System.currentTimeMillis() - before;
   ```

2. Recall the searching algorithms from lecture.

   (a) First, write the generic static `myBinarySearch` method, which receives the list and the key to search. The method should return the index of the string in the list, or -1 if it does not exist.

   (b) Output (using `System.out.println`) the time difference, in milliseconds, between your version of binary search and Java's `Collections.binarySearch` method by searching for the string `"computer"`. How big (or small) of a difference is there?