CS 162C Self Evaluation for Lab 2 – Searching and Sorting Parallel Lists

Your name: Joseph Sepe	Date: 4/12/2021
Are you willing to allow your code to be used in example debugging demonstrations or documentation? [✓ Yes] No	

Instructions – Part 1

This document is to be turned in alongside solution of this lab. You will use this document to indicate your status on the lab, as well as areas where you are struggling conceptually or in converting concept to code. Please use the space underneath each evaluation criteria to describe any errors you are receiving or challenges you are having implementing the required functionality for your code.

Functionality

Basic Expectations	***Completed***
Does the program compile and run?	Yes
Does the program run correctly and return the expected result?	Yes
Are there comments explaining what the program and various functions are doing?	Yes
Functions	***Completed***
Are all required functions implemented?	Yes
Does the fillLists function correctly sort items as they are added into the lists?	Yes
Does the fillLists function correctly insert values into both lists together?	Yes
Does the fillLists function stop after reaching the correct (user input) number of values?	Yes
Does the sortLists function properly implement a selection sort?	Yes
Does the sortLists function properly stay within the pseudo array bounds?	Yes
Does the displayLists function properly format and display each list in its own column?	Yes
Do you properly implement a binary search to find values in your lists?	Yes
I'm only doing a binary search on the names list. I didn't see instructions for searching for Lnumber	

<u>Instructions – Part 2</u>

Please answer the following questions, in your own words, regarding your experiences throughout this lab.

Experiential Review

What aspects of this lab did you find most challenging?	
Sorting without the use of the built in list functions. The built in list functions are just so easy to use	
What concept from this lab do you feel you have the best grasp on now?	
Swapping values of an array. Working with indexes.	
Please explain the effectiveness of linear and binary searches on both ordered and unordered lists/arrays.	
A binary search will be most effective when the values it's searching for is towards the middle indices on each split. If the list is sorted it will search by value, if it isn't sorted you'll have to search by each index. A linear search isn't affected by the order of the array, it just goes in a line. If the value you're searching for is at the end of the line it will be slower though. For example if it's ordered 1-10 and you're searching for 10, it will take you 10 checks before you get it, but if it's an unordered list of 1-10 you might find 10 quicker.	
Please compare the three sorts and explain why you should or should not use each.	
A bubble sort will make multiple passes through the list, look at two adjacent values and then swap them if needed, then move on. It's a slow process and maybe only use this if the data is mostly sorted already. Selection sort goes through the list and moves the largest to the top of the array, then repeats continuing with moving the next largest value. You'd want to use it for finding the smallest or largest value quickly. Insertion sort is when it takes a value in the list and then finds where it's supposed to go and then moves the location of everything up and inserts it where it belongs. It doesn't do swaps, it actually pushes other values around and it is the most efficient of all the sorts especially for mostly sorted data.	
What are parallel lists? What are the most important things to keep in mind when implementing them in your code?	
Parallel lists are two lists with corresponding values in their indices. You want to keep in mind that if you change the location of one value in a list, you have to change the location of the other value in the other list. If a value gets added or deleted, it has to have a corresponding value added or deleted in the same index. You also want to keep them the same size.	
How do you think you could modify a sort to implement a shuffle instead? Which sort do you think would be best for this and why?	
I would implement it maybe by adding a random number and swap random indices with each pass. I'd maybe use a selection sort because it is not stable and remove the part where you find the index of the next value and just swap random values.	