COMP108 Data Structures and Algorithms
Week 05 Lab Exercises

Due: 3 March 2023, 5:00pm

(Late submission accepted until Monday 9:00am)

Information

- Submission: Submit the file **COMP108W05.java** on Canvas
  Late submission is only accepted until Monday 9:00am.
- Submission of lab/tutorial exercises contributes to 10% of the overall module mark. Submission is marked on a pass/fail basis - you will get full marks for submitting a *reasonable attempt*.
- Individual feedback will not be given, but solutions will be posted promptly after the deadline has passed.
- These exercises aim to give you practices on the materials taught during lectures and provide guidance towards assignments.
- Relevant lectures: Lectures 2-6

1. **Programming — Preparation**

   You have been asked to prepare your programming environment in Week 1. If you haven't done that yet, follow the discussion on Canvas:

   https://liverpool.instructure.com/courses/61186/pages/compiling-and-running-java-programs

   You can use web IDE: https://ide.cs50.io/ if you haven't setup your own environment.

   (a) Download the following java files "COMP108W05App.java", "COMP108W05.java" from Canvas via the link "Labs & Tutorials".

   (b) Open the files with a text editor (e.g., notepad++ or web IDE editor).
   Beware of where you have saved the java file and open it from the correct folder. Do NOT use MS Word!

   (c) COMP108W05App.java takes care of data input. The algorithms to be implemented are in the file COMP108W05.java.

   (d) Open a command prompt (cmd) and change to the folder where you saved the programs.

   (e) Refer to instructions in Week 04 on how to compile the programs.

2. **COMP108W05App.java and COMP108W05.java**

   (a) **Background**

   i. In COMP108W05.java, some methods are already implemented: `printArray()`, `seqSearch()` and `findMin()`. You are going to implement a number of searching algorithms.

   ii. For simplicity, the programs have been hard-coded so that the array contains 20 integers. Try finding some integers that are in the array and some that are not in the array.
   The array contains 15, 25, 10, 30, 35, 20, 5, 60, 80, 65, 75, 70, 100, 55, 90, 45, 50, 85, 95, 40
   The sorted array contains 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100

   (b) **Task 1 — Binary Search**

   i. The sequential search algorithm has been implemented in the method **seqSearch()** to check if *key* is in the array data[ ] of size n. The method also counts how many comparisons are made.

   ii. Implement in the method **binarySearch()** of COMP108W05.java the **binary search** algorithm to determine if *key* is in the array data[ ] of size n.
   You can refer to the pseudo code in lecture notes.
   Mind that the lecture notes assume numbers are stored in A[1], A[2], $\cdots$, A[n] but the program assumes numbers are stored in data[0], data[1], $\cdots$, data[n-1], which are already sorted in ascending order.

   iii. Compile, run, and test your program to see if it determines correctly whether a number is in the array.

   iv. Update the binarySearch() method to also count the number of comparisons, i.e., it should also count how many values in the array has been compared with.

   v. The number of comparisons should be between 1 and 5. If you follow the logic in the lecture notes, the output should be:

input	# comp
50	1
25 or 75	2
10, 35, 60 or 90	3
5, 15, 30, 40, 55, 65, 80 or 95	4
20, 45, 70, 85, or 100	5
anything else	4 or 5

   (c) **Task 2 — Finding max/min**

   i. Read the method **findMin()** to understand its logic.

   ii. Fill in the method **findMax()** to find the largest number. It should look very similar to the find_min() method.
   Test if it outputs the largest number in the array hard-coded in the program.

   iii. The method **findSecondMax()** aims to find both the largest and the second largest numbers in the array.
   Fill in **findSecondMax()** the code to do so.