COMP108 Data Structures and Algorithms
Lab Exercises (Week 3)

Due: 26 February 2021, 5:00pm

(Late submission not allowed)

Information

- Submission: Submit the file COMP108W03.java to SAM https://sam.csc.liv.ac.uk/COMP/CW_Submissions.pl?qryAssignment=COMP108-13
- 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*.
- Late submission is **NOT** possible. 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: Lecture 6 (Video 2) & Lecture 8 (Video 1)
- You can refer to the guidance on how to use the web-based IDE https://ide.cs50.io/.

1. **Programming — Preparation**

   (a) Download two java files "COMP108W03App.java" and "COMP108W03.java" from Canvas via the link "Labs & Tutorials" → "Week 3".

   (b) Compile the programs by typing first **javac COMP108W03.java** and then **javac COMP108W03App.java**. There should be two files created: COMP108W03.class and COMP108W03App.class.

   (c) Run the program by typing **java COMP108W03App**.

   (d) **Every time you have edited COMP108W03.java, you have to (i) recompile by javac COMP108W03.java and then (ii) run by java COMP108W03App.**

   (e) You are going to work on COMP108W03.java in which some methods are already implemented: `printArray()`, `seqSearch()` and `findMin()`. You are going to complete five methods: `binarySearch()` for Task 1; `findMax()`, `findSecondMin()` and `findSecondMax()` for Task 2; `bugOne()` for Task 3.

   (f) For simplicity, the program does not validate input. Note that 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. Only the sequential search algorithm has been programed.

   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

2. **Programming with Arrays Task 1 — Searching**

   This question follows the question in the last lab session to help you get familiar with programming loops and extend to use arrays.

   (a) **Enter your name and student ID** to the beginning of COMP108W03.java.

   (b) 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 used.

   (c) Implement in the method **binarySearch()** of COMP108W03.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 6.

   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.

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

   (e) 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.

   (f) 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

3. **[Programming with Arrays Task 2 — Finding max/min]**

   This question continues to work on arrays and asks you to fill in THREE methods.

   Only the method **findMin()** has been completed. The expectation of the final output would include: finding the smallest, second smallest, largest and second largest number.

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

   (b) 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.

   (c) The method **findSecondMin()** is supposed to find both the smallest and the second smallest numbers in the array.

   Fill in **findSecondMin()** the code to do so.

   You can refer to the pseudo code in Lecture 8.

   Test if it outputs the smallest and the second smallest numbers correctly. Compare your output of the method findMin().

(d) [*Optional: Do this if you have time as this should really be combining what you have done above.*]

Similarly fill in **findSecondMax()** which aims to find both the largest and the second largest numbers in the array.

4. **Optional Programming Task - Fixing bugs** [do this if you finish Tasks 1 and 2]

   (a) Continue to work on COMP108W03.java in particular the method `bugOne()`

   (b) The method bugOne() attempts to find the smallest number and its position in a given array. Currently, it does not output the correct value. The method can be fixed by altering **ONE** line of code. Try to fix it if possible by altering only one line of code.