COMP108 Data Structures and Algorithms
Week 08 Lab Exercises

Due: 24 March 2023, 5:00pm

(Late submission accepted until Monday 9:00am)

Information

- Submission: Submit the file COMP108W08.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 10 & 11**

1. **Programming — Preparation**

   (a) Download three java files "COMP108W08App.java", "COMP108W08.java" and "Node.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) COMP108W08App.java takes care of data input. The algorithms to be implemented are in the file COMP108W08.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. **Linked List** This week we will work with linked list moving nodes around.

   - A class `Node` is defined in Node.java to have three attributes: `data`, `next`, `prev`.
   - A class `COMP108W08` is defined in COMP108W08.java to have two attributes: `head`, `tail`, which point to the head and tail of a list.
   - Several auxiliary methods have been implemented, including
     `public void insertHead(Node newNode)` insert newNode to head of list;
     `public void insertTail(Node newNode)` insert newNode to tail of list;
     `public String headToTail()` goes through the list and return a String containing the data of each element of the list from head to tail;
     `public String tailToHead()` goes through the list and return a String containing the data of each element of the list from tail to head;

- The methods insertHead() and insertTail() are written to create the list (and as an illustration). You don't have to use them for your tasks and it's probably easier if you don't use them.
- The methods headToTail() and tailToHead() are meant only to convert the list to String for easy display. Do not use them to carry out your tasks.

Study these methods to see how to go through the linked list.

3. **Task 1: sequential search and move to head**

   The method searchMoveToHead() takes a parameter key and aims to (i) find if key exists in the list, (ii) if yes, move the node containing key to the head of the list; if no, nothing needs to be done.

   You can refer to Week 07 Lab for how to do sequential search.

   Complete the method (without changing its signature) and test it using test cases stated at the end of the document.

   Remarks: You are expected to go through the list using sequential search on list. Do not convert the list into an array to process it. Also, do not use the split method of the String class or the parseInt method of the Integer class to work on the String returned by headToTail() or tailToHead(). The latter two methods are only meant to ease printing from the list to help debugging.

4. **Task 2: Sequential search and move to tail**

   The method searchMoveToTail() takes a parameter key and aims to (i) find if key exists in the list, (ii) if yes, move the node containing key to the tail of the list; if no, nothing needs to be done.

   Complete the method (without changing its signature) and test it using test cases stated at the end of the document.

   The remarks in Task 1 also applies here.

5. **Test cases:**

input			output			
# of int.	input integers	key	searchMoveToHead		searchMoveToTail	
			From head	From tail	From head	From tail
3	0 10 -10	10	10 0 -10	-10 0 10	0 -10 10	10 -10 0
5	10 20 30 20 40	10	10 20 30 20 40	40 20 30 20 10	20 30 20 40 10	10 40 20 30 20
5	10 20 30 20 40	40	40 10 20 30 20	20 30 20 10 40	10 20 30 20 40	40 20 30 20 10
5	10 20 30 20 40	20	20 10 30 20 40	40 20 30 10 20	10 30 20 40 20	20 40 20 30 10
2	10 20	10	10 20	20 10	20 10	10 20
1	10	10	10	10	10	10