ITI 1121. Introduction to Computing II

Assignment 4
Submission **deadline**: See **Brightpsace**

Assignment: Group of 2 students Maximum.
Submission: A3_123456_123789.zip the two numbers are the students' numbers.

Reminders: The submitted code must be **yours**.
You are also responsible to **not** expose your code to others.

The submission must be done through Brightspace, **before the deadline**.

- You must test your programs to make sure they give the correct results.
- A program that gives a **compilation error or an execution error will receive a mark of zero.**
- For this assignment, there are no marks associated to exception handling, however, your program shouldn't crash at execution time.
- Do not import any extra packages.
- You have to put **comments** in the programs. The comments are **not** needed for each line of code. However, they have to explain in **simple English** what the program does.
- Some java files are provided, **they should be used** and the content already there **cannot** be modified. Which means you need to **keep** the existing code and **add** your code to it.
- Only submit your **java** files, do **not** submit any .class files.
- Important: one submission per group is enough, however both students are responsible to make sure the submission was done before the deadline.

Academic Integrity

https://www.uottawa.ca/administration-and-governance/academic-regulation-14-other-important-information
https://www.uottawa.ca/vice-president-academic/academic-integrity

By submitting this assignment, you acknowledge the following:
1. I have read the academic regulations of the University of Ottawa regarding academic fraud.
2. I understand the consequences of plagiarism.
3. With the exception of the source code provided by the professor for this course, all the source code is mine and my group partner only.

A4Q1

Implement the method isComprisedBetween(E v1, E v2) for the elements of a singly linked list myList. The method returns the reference to a new singly linked list. The elements of the new linked list have a value between v1 and v2, inclusively (value>=v1 and value<=v2).
The class of the singly linked list implements the interface Comparable<E>.
Use the method compareTo (no need to impletement it) to compare the elements. The call a.compareTo(b) returns:

- a negative integer if the value of a is less than b.
- zero if a and b are equal.
- a positive integer if the value of a is greater than b.

For this to work, you need to first reorder the list, myList. So the method isComprisedBetween must first call the method orderList() that will return a new list which is the sorted version of myList in ascending order. You cannot use another structure to complete this question (such as arrays or other), you need to use the LinkedList structure only.
Implement the method orderList() which returns a new sorted LinkedList. myList will remain unchanged.

Example: given the linked list, myList, containing the values 7, 5, 3, 9, 1, 4, 2, 6, 8, a call to:

myList.isComprisedBetween(5,7) returns the linked list of values 5,6,7. myList is unchanged at the end of the execution.

- The methods names have to be exactly as in the template given, do not change them.
- The result list should be in ascending order. You do not sort it, you just add the elements in order.
- You are **not allowed** to use the methods addFirst or addLast or removeFirst or removeLast or any other such methods to modify the lists. You need to add/remove elements directly within either isComprisedBetween(E v1, E v2) or orderList(), by accessing any needed elements using the operator .next. (addLast etc can only be used to develop tests within UseList.java)

Develop multiple tests within the java file UseList.java to test your program. Include different cases, at least include tests for empty list, list of one element, list of multiple elements, testing with elements v1 and v2 that don't exist in the list, testing with an already ordered list.

Test example:

```
testCase1: result for myList.isComprisedBetween(2,3)

initial list: [1, 2, 3, 4, 5, 6, 7, 8, 9]

new list: [2, 3]

initial list unchanged: [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

A4Q2

In this question you will use doubly linked lists. Implement the method void addSpecific(E beforeMe, DoublyLinkedList<E> someList). When this method is called for a list myList:

- it adds the last element of someList before the last occurrence of beforeMe in myList.
- and that element that was added to myList is removed from someList.

You cannot use another structure to complete this question (such as arrays or other), you need to use the DoublyLinkedList structure only.

If beforeMe element is not found in myList, print a message "specific element not found", myList and someList remain unchanged.

- The methods names have to be exactly as in the template given, do not change them.
- You **cannot** add any instance variables. The only instance variables are head and size.
- You are **not allowed** to use the methods addFirst or addLast or removeFirst or removeLast or any other such methods to modify the lists. You need to add/remove elements directly within the addSpecific method. (addLast etc. can only be used to develop tests within UseDoubleList.java)

Develop multiple tests within the java file UseDoublyLinkedList.java to test your program. Include different cases. At least include tests for empty list, testing when beforeMe doesn't exist in myList, testing one occurrence of beforeMe in myList, and multiple occurences of beforeMe in myList.

Example: Suppose the lists contain the following values:

- myList has the values ["ITI1121" ,"Hello","ITI1121","Summer"],
- and the list someList has the values["Hi","AddMe"].

The call: myList.addSpecific ("ITI1121", someList] will change the lists as follows:

- ➔ myList has the values ["ITI1121" ,"Hello", "AddMe","ITI1121","Summer"]
- ➔ and someList has the remaining value(s) ["Hi"].