CSC 212 Homework # 2
ADT List & Double Linked List
Due date: 28/10/2017

Guidelines:	This is an individual assignment. The homework must be **submitted electronically through LMS**. **Submissions by email and hard copy submissions are not accepted**.

Problem 1

1. Write the method **public** `<T>` **static void** `clear(ArrayList<T> l)` which removes all the elements of l. What would you need to change if the list `l` were of type `LinkedList`?

2. Write method `insertAll` as user of ADT List that takes two lists `l1`, `l2` and index `i` and insert all elements in `l2` in `l1` after position `i`. The list `l2` must not be changed. The first element has position 0, and assume that `i` is a valid position.

   Example 1.1. *If* $l1 : A, B, C, D$, *and* $l2 : X, Z$, *then after calling* `insertList( l1, l2, 1)`, *then* $l1 : A, B, X, Z, C, D$.

   Method signature **public** `<T>` **void** `insertAll(List<T> l1, List<T> l2,` **int** `i)`.

3. Write the method: **public static** `<T>` **void** `commonE(List<T> l1, List<T> l2, List <T> cl)`, user of the ADT List, which inserts the common elements between list *l1* and list *l2* in the list cl. Assume that the elements in *l1* and *l2* are unique and the List *cl* is initially empty.

   Example 1.2. *If* $l1 : A, B, C, F, M, D$, *and* $l2 : R, M, W, F$, *calling* commonE(l1, l2, cl) *results in* $cl : F, M$

. . . . . .

Problem 2

1. Implement the following methods in the class `LinkedList`:

   (a) **Procedure** insertBeforeCurrent(T e). **Requires**: The list l should not be full. **Results**: The new element e is inserted before the current and the new element is made the current.

   (b) **Procedure** removeIth(int i). **Requires**: The list l should not be empty. **Results**: The element e at position i is removed from the list (numbering starts with 0), If the resulting list is empty current is set to NULL. If successor of the deleted element exists it is made the new current element otherwise first element is made the new current element.

2. Write the method `removeEvenElems`, member of the class `ArrayList`, that removes all the elements having an even position (the position of the first element is 0). The method must run in $O(n)$. The method signature is: **public void** `removeEvenElems ()`. **Do not call any methods and do not use any auxiliary data structure**.

   Example 2.1. *If* $l : A, B, C, D, E$*, then after calling the method* `l.removeEvenElems ()` l *becomes:* B, D*.*

. . . . . .

Problem 3

1. Write the method `checkListEndsSymmetry` that receives a double linked list and an integer number k. The method checks if the double linked list has identical k elements going forward from the first element and backwards from the last one. The method returns *true* if they are identical, and *false* otherwise. The method signature is: **public** `<T>` **boolean** `checkListEndsSymmetry(DoubleLinkedList<T> dl,` **int** `k)`

   Example 3.1. *If* $dl = A \leftrightarrow B \leftrightarrow C \leftrightarrow D \leftrightarrow B \leftrightarrow A$ *and* $k = 2$*, then the method should return* true*. If* $k = 3$*, it should return* false*, since* C *does not equal* D*.*

2. Write the method *bubbleSort* that sorts a double linked list of integers given as input using bubble sort. The method signature is: **public void** `bubbleSort( DoubleLinkedList<Integer> l)`.

. . . . . .

Problem 4

One of the limitations of the array implementation of the ADT List seen in class is the size limit. It is, however, possible to solve this problem through a dynamic reallocation of the array as follows.

- The size of the array is initially set to a small value (say 1).
- If the capacity of the array becomes insufficient, a new array is allocated having double the size of the current one. The data is then transferred from the old array to the new one, and the old array is discarded.
- If percentage of used space drops below a given value (for example 40%), a new array is allocated having half the size of the current one. The data is then transferred from the old array to the new one, and the old array is discarded.

Complete the implementation of the class `DArrayList` that uses this technique: Write the methods: `full`, `insert` and `remove`.

```
public class DArrayList<T> implements List<T> {
        private T[] data;
        private int current, size, maxSize;
        private static final double minRatio = 0.4;
        public DArrayList() {
                data = (T[]) new Object[1];
                maxSize = 1;
                current = -1;
                size = 0;
        }
}
```

. . . . . .

Problem 5

A circular list is a list with no first or last element and where the current advances in a circular way through the data.

1. Give a clear specification of the ADT *CircularList* (use the same format seen in lecture).
2. Write down the interface `CircularList`.
3. Write the method **public void** `print(CircularList<String> l)`, which prints the content of `l` starting from the current element. At the end of the method, current must return to its initial position.
4. Give a linked implementation of the interface `CircularList` (class `LinkedCircularList`).

. . . . . .