Sorting a list (7 marks)

- Don't forget to set your Eclipse workspace and working set.
- You must submit the JAR file, exported (with source code), from your Eclipse project.
- You must check your JAR file to make sure all the source files (.java files) are present. It can be opened with file compression programs such as 7-zip or Winrar.
- Failure to export properly will result in your work not getting marked.

To submit:

- Export your project to a JAR file, with source code.
- Name your JAR file ID_Week13_Q1.jar. For example, 6623110021_Week13_Q1.jar
- Submit the JAR file on MyCourseville.

(7 marks, will be scaled to the same score as other homework)
You are given classes for Circular doubly linked list. Your task is to write method

```
public void sort() throws Exception { of class CDLinkedList
```

This method sorts numbers stored in 'this' list from small to large.

- You must use quick sort algorithm, implement it on the linked list.
  - sort() must call public void quicksort(DListIterator start, DListIterator finish)
    - start marks the first data in the portion to be sorted.
    - Finish marks the last data in the portion to be sorted.
  - The pivot is the last data in the portion that is to be sorted.
  - There is no call to other sorting algorithm. The quick sort is done regardless of the list size.
  - You can write a method to test whether an iterator is to the left (regarding the position in the list) of another iterator. This method is allowed to run in O(n). **Be very careful when calling next() or previous().**
- Array is not allowed to be created or used.
- You are not allowed to create any other data structures. Even a linked list must not be created.
- You can only use 'this' as your only data structure.
- JUnit Test cases used when marking will be different from the JUnit you have, but they test the same logic. So make sure your program works for any possible inputs.

  Guide for revision (after you finished the question):
  - Write your method several times, each time with different sorting algorithm.
  - Try to write bubble sort, selection sort, insertion sort in < 10 minutes.
  - Try to write merge sort, quick sort in < 15 minutes.
  - You only need to submit the quick sort version.

Score Criteria:
The code must use quick sort, if not, you get 0 mark.

- testSortEmptyList() 1 point
- testSortListOneData() 1 point
- testSortListEvenData() 1 point
- testSortListEvenDataReverse() 1 point
- testSortListOddData() 1 point
- testSortListOddDataReverse() 1 point
- testSortListAlreadySorted() 1 point