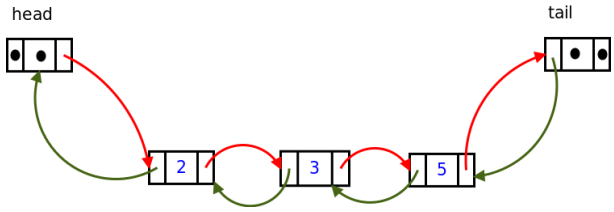
Assignment 4

due 27/3/2015

This is assignment involves the implementation in Java of the set abstraction using linked lists.

1. Proceed as follows.

   - Download the file `a4.tgz` which contains some material you will need.
   - Make sure you study the ADT description carefully before attempting to touch the code.
   - You will only need to write LinkedSet.java by fleshing out the stub provided.
   - Your implementation must be based on the doubly-linked list concept. So the set $\{2, 3, 5\}$ would be represented as follows.



   - Keep your work within the a4 directory. Submit all your work (as a4.tgz) using Moodle on or before the deadline stated above.

ADT Set

2. Recall that the mathematical concept of a *set* is a collection of distinct *elements*, *e.g.* α, β, γ. ADT Set will allow sets to be represented and manipulated by a variety of operations (outlined below).

3. The basic operations supported by ADT Set are as follows. Notice that some of these operations take arguments of type Set, others return results of type Set. The place holder E is used to designate the type of element that the set contains [1]. Make sure you understand these descriptions thoroughly before attempting to write any code. *In particular note that the specification of a set precludes duplicate elements.*

   - **isEmpty():** Return true if this set contains no elements. *Input:* None; *Output:* boolean.
   - **size():** Return the number of elements in this set (i.e. its cardinality). *Input:* None; *Output:* int.

[1] While the mathematical notion of a set does not preclude sets that contains elements of differing types, our formulation of ADT Set is restricted to homogeneous sets in which all the elements have the same type. Nor does ADT Set permit sets to be infinite, for obvious reasons.

- **add(newElement):** Add the specified element to this set if it is not already present. If this set already contains the specified element, the call leaves this set unchanged *Input:* E; *Output:* None.
- **contains(checkElement):** Return true if this set contains the specified element i.e. if checkElement is a member of this set. *Input:* E; *Output:* boolean.
- **remove(remElement):** Remove the specified element from this set if it is present. *Input:* E; *Output:* None.

4. The ADT also supports the following iterator method.

   - **elements():** Return an iterator of the elements in this set. The elements are returned in no particular order. *Input:* None; *Output:* Iterator<E>.

5. In addition the ADT supports a number of additional methods that operate on pairs of sets.

   - **addAll(addSet):** Add all of the elements in the set addSet to this set if the are not already present. The addAll operation effectively modifies this set so that its new value is the union of the two sets. *Input:* Set<E>; *Output:* None.
   - **containsAll(checkSet):** Return true if this set contains all of the elements of the specified set i.e. returns true if checkSet is a subset of this set. *Input:* Set<E>; *Output:* boolean.
   - **removeAll(remSet):** Remove from this set all of its elements that are contained in the specified set. This operation effectively modifies this set so that its new value is the (asymmetric) set difference of the two sets. *Input:* Set<E>; *Output:* None.
   - **retainAll(retSet):** Retain only the elements in this set that are contained in the specified set. This operation effectively modifies this set so that its new value is the intersection of the two sets. *Input:* Set<E>; *Output:* None.

Implementation Notes

6. As linked structures can often be frustrating to work with at first, it is important to tackle this assignment in good time that you approach it methodically.

7. Tackle the "easy bits" first. Implement the zero-argument constructor (*i.e.* the one that does not take a comparator) and the methods add, contains and remove.

8. When implementing the iterator, review the material in Lecture 11 that deals with iterator implementation (albeit in the context of ADT List). The same "snapshot" technique employed there using ArrayBasedIterator will work here.

9. The ∗All methods are easy to implement if you go about it sensibly.