

Exercise 9: Using linked lists

The files for this exercise can be found in the directory

<http://www.dcs.qmul.ac.uk/~mmh/DCS128/exercises/linkedlists>

Before doing this exercise, you might find it useful to look at some additional exercises you can find in the file `CellsAndPointersExercise.html` in this directory.

Exercise 9 is intended to get you used to programming with linked lists. A program which will help you get started can be found in the file `UseLinkedLists.java`.

This program provides you with a front-end that enables you to read lists of integers from the screen. The string representation of a list it expects starts with the character `[`, ends with a matching `]`, and has the numbers separated by commas. So, for example, `[7,12,5]` represents a list of three integers, the first 7, the second 12 and the third 5. The method called `parseIntLinkedList` takes a string and returns the linked list of integers this string represents. The method `linkedListToString` does the reverse. But just use these, do not be concerned with how they work or attempt to modify them. Note that although we used the same textual representation for Lisp lists, the two are slightly different concepts (“Lisp list” is an abstract data type, “linked list” is a data structure).

Two methods in the file `UseLinkedLists.java` are given as examples of the sort of code you should be writing for this exercise. The methods are `destChange` and `constChange`. Both are generic methods which take two arguments of the type of its type variable, and the third argument is a linked list of this base type. Both have the effect of changing all occurrences of the first argument to the second argument in the linked list, but `destChange` does this destructively, while `constChange` does it constructively (that is, constructs a new linked list representing the change).

In `UseLinkedLists.java`, the operations are implemented using iteration. Linked list operations may often be conveniently implemented using recursion. For comparison, the same operations are shown implemented using recursion in the file `UseLinkedListsRec.java` in the same directory. Further examples of code for operations on linked lists can be found in section 19 of the notes on “Implementing ArrayLists using linked lists”, where a linked list is used as the data structure inside an object implementing the Abstract Data Type `ArrayList`.

Your task in this exercise is to write some more methods involving linked lists. Where the methods involve changing linked lists, you should try to write both destructive and constructive versions. Here are some problems to try:

- 1) Test whether a particular item is in a linked list. So if the item is 4 and the list is `[1,4,2,5]`, the method returns `true`; if the item is 6 and the list is `[1,4,2,5]`, the method returns `false`.
- 2) Delete the first occurrence of an item from a linked list. So if the item is 7 and the list is `[1,3,7,4,3,7,2]`, the result is `[1,3,4,3,7,2]`.
- 3) Delete all occurrences of an item from a linked list. So if the item is 7 and the list is `[1,3,7,4,3,7,2]`, the result is `[1,3,4,3,2]`.

- 4) Delete the last occurrence of an item from a linked list. So if the item is 7 and the list is [1, 3, 7, 4, 3, 7, 2], the result is [1, 3, 7, 4, 3, 2].
- 5) Count the number of times a particular item occurs in a linked list. So if the item is 7 and the list is [1, 3, 7, 4, 3, 7, 2], the result is 2.
- 6) Given an item and a linked list, return the position of the first occurrence of the item in the linked list, or -1 if it does not occur. So if the item is 5 and the list is [2, 4, 5, 8, 1, 5, 3], the result is 2 (using the Java convention that positions start at 0).
- 7) Given an item and a linked list, return a list giving the positions of all occurrences of the item in the list argument. So if the item is 5 and the list is [2, 4, 5, 8, 1, 5, 3], the result is [2, 5].
- 8) Given two linked lists, join them together to give one. So if the lists are [1, 3, 7, 4] and [2, 4, 5, 8, 1], the result is [1, 3, 7, 4, 2, 4, 5, 8, 1].
- 9) Given a linked list of numbers and a number, add the number to all the numbers in the list, so if the number is 20 and the list is [1, 5, 12, 9, 7, 16], the result is [21, 25, 32, 29, 27, 36].
- 10) Given a linked list and two items, insert the second item after every occurrence of the first item in the list. So if the list is [2, 4, 3, 2, 8, 2, 5, 1, 2] and the items are 2 and 10, the result is [2, 10, 4, 3, 2, 10, 8, 2, 10, 5, 1, 2, 10].
- 11) Test whether a linked list of numbers is in ascending numerical order, for example [1, 5, 9, 12] is, but [1, 5, 12, 9] is not.
- 12) Merge two linked lists of numbers which are in ascending numerical order, so if the lists are [2, 3, 4, 8, 12] and [1, 4, 5, 7, 9], the result is [1, 2, 3, 4, 4, 5, 7, 8, 9, 12].
- 13) Merge two linked lists of numbers, but if a number occurs in both lists, only include it once in the final list. So if the lists are [1, 3, 4, 7, 12] and [1, 5, 7, 9], the result is [1, 3, 4, 5, 7, 9, 12].
- 14) Merge two linked lists of numbers, including in the final list only those numbers that occur in both lists. So if the lists are [1, 3, 4, 7, 12] and [1, 5, 7, 9], the result is [1, 7].

Note that while the examples given here show linked lists of numbers, only some of the questions rely on the base type being numerical. So for the others, you should be able to write generic code which deals with linked lists of any base type.

Questions 11 to 14 mention “numerical order”, but a generic solution could deal with any type of object which has a natural ordering, for examples strings with their alphabetic ordering. You would then need to use a bounded type variable (as covered in section 17 of the notes) to write generic methods that could deal with linked lists of integers or linked lists of strings. To start you off, the file `UseLinkedListStrings.java`, in the same directory as the other files mentioned previously, shows generic methods which insert an item into an ordered linked list of items, demonstrated using linked lists of strings ordered alphabetically. A method for destructive insert and a method for constructive insert is given. The original ordered list is obtained by calling Java’s `sort` method on the words when they are initially entered and stored in an array before they are put into a linked list.