

Homework 4: Linked Lists

cs230
Spring 2002

Allen B. Downey
Computer Science Department

Due: Tuesday 5 March

The purpose of this assignment is to implement some of the methods that operate on lists. There are three goals for this exercise: practice with manipulating embedded references, thinking about interface design issues, and building infrastructure for future assignments that will use lists to implement other data structures. Before starting, you should read Chapter 14.

Set up `LinkedList`

1. Create a new directory named `hw04`.
2. Download the files `LinkedList.java`, `Node.java` and `Test.java` from <http://rocky.wellesley.edu/cs230/code/hw04/>.
3. Compile and run the program. It should look like this:

```
% javac Test.java; java Test
(1, 2, 3)
(3, 2, 1)
```

Add more methods to `LinkedList`

The following methods are all object methods in the `LinkedList` class, but you might want to write some helper methods in the `Node` class as well.

After you write each method, add code to `main` in `Test.java` and test it. Be sure to test special cases like empty lists and singletons. Try to write each method in a way that makes it obvious that the method is correct.

Also, for each method, identify any preconditions that are necessary for the method to work and add comments that document them. Your comments should also indicate whether each method is a constructor, function, or modifier.

1. Write a method named `removeFirst` that removes the first node from a list and returns its cargo.
2. Write a method named `set` that takes an index, `i`, and an item of cargo, and that replaces the cargo of the `i`th node with the given cargo.
3. Write a method named `add` that takes an index, `i`, and an item of cargo, and that adds a new node containing the given cargo in the `i`th position.
4. Write a method called `clone` that returns a new `LinkedList` that contains all the same cargo as the original list, but it should contain all new `Node` objects. You might want to write a helper method in `Node` to copy the nodes and link up the copies. You might find it natural to write that method recursively.

5. Write a method called `reverse` that modifies a `LinkedList`, reversing the order of the nodes.
6. Write a method named `addLast` that takes an item of cargo and adds it to the end of the list.
7. Write a method named `append` that takes a `LinkedList` as a parameter and appends a *copy* of the nodes from the parameter list onto the current list. You should be able to take advantage of code you have already written.