Algorithms & Data Structures: Lab 04

week of 22nd October 2018

1 Setup

1.1 Saving your work from last week

As with previous weeks, you will use `git` to download a bundle of lab code. You might have made modifications in your downloaded copy; if you have not already done so, you need to save those modifications. First examine the changes present in your downloaded copy by issuing the following commands from the labs directory:

```
git status
git diff
```

and if you are satisfied with the changes, store them in the git version control system by doing

```
git commit -a
```

and writing a suitable commit message

1.2 Downloading this week's distribution

Once you have successfully saved your changes from last week, you can get my updates by doing

```
git pull
```

which *should* automatically merge in new content. After the `git pull` command, you should have a new directory containing this week's material (named `04/`) alongside the existing directories.

2 Linked Lists

2.1 Basic implementation

Implement a `SLList` linked-list class, whose basic methods are: `first()`, `rest()`, `setFirst()`, and `setRest()`. The class must be able to store *at least* the default integer range in your programming language; the "rest" must be a reference or pointer to another `SLList`, including the special `SLList` object `NIL`.

You should also provide a two-argument constructor, which initializes the instance with the first argument as the first, and the second argument as the rest, of the resulting `SLList`.

I have provided skeleton code under `04/` to help you structure your work, and tests for this functionality which can be run using `make test` as usual.

2.2 Derived methods

Extend your implementation of linked lists to support three additional methods, which you should be able to implement in terms of the existing ones or directly:

`nth()` return the n$^{\text{th}}$ item stored in the list, counting from 0 (so `nth(0)` should return the first item)

`nthRest()` return the n$^{\text{th}}$ rest of the list, counting from 0 (so `nthRest(0)` should return the given list itself)

`length()` return the length of the list.

I have provided tests for `nth()` and `nthRest()`; you are responsible for testing your own implementation of `length()`. How can you assure yourself that your implementation is correct? How can you convince the person working next to you?

2.3 More derived methods

You might wish to work through the interactive exercise related to recursively-expressed algorithms on linked lists, available from the module VLE page.

2.4 Submission

There will be a submission related to this lab in two weeks (deadline **16:00 9th November 2018**); you will be asked to submit work based on your implementation of the `SLList` class. Make sure you save your work, and that you understand what is going on.