Algorithms & Data Structures: Lab 05

week of 29th 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 `05/`) alongside the existing directories.

2 Linked Lists (cont'd)

2.1 Setup

You should begin from the work you did in lab 04: a singly-linked list class implemented in terms of `SLList` objects, with primitive methods:

- `first()`
- `rest()`
- `setFirst(o)`
- `setRest(r)`

and additional methods:

- `nth(i)`
- `nthRest(i)`
- `length()`

Verify that your implementation compiles and runs the additional tests provided in this week's distribution.

2.2 Remove

Implement `remove`, which returns a new list whose contents are the same as the original list, but with all instances of the given object removed. Use the second part of the interactive exercise on list algorithms to guide you, and check your implementation against the provided test cases.

2.3 Reverse

Implement `reverse`, which returns a new list whose contents are the same as the original list but in reverse order. Check your implementation against the provided test cases. What are the time and space complexity of your solution?

2.4 Other linear list algorithms

Write down the base case and recursive step for the following algorithms:

sum find the sum of the elements in a list

max find the maximum value of the elements in a non-empty list

search[o] return true if the object is present in a list, otherwise false

position[o] return the position of the object in a list, if present, otherwise -1

By adding to your existing `SLList` implementation, write methods to perform these operations. How will you test your implementations?

2.5 Submission

Submit your work on the `SLList` data structure and related algorithms to the lab submission area on the module learn.gold page. The submission area will close at **16:00** on **9th November 2018**.