Algorithms & Data Structures: Lab 07

week of 19th November 2018

1 Setup

1.1 Saving your work from last week

As with all previous weeks, you will use `git` to download a bundle of lab code. You will probably 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 `07/`) alongside the existing directories.

2 List visualiser

2.1 Implementing a class to visualise a list

You have (in labs 04-06) already implemented a singly-linked list class in terms of `SLList` objects. For this part of the lab, you must implement a method to visualise a linked list object.

Specifically, you must implement a `ListVisualiser` class, with:

- a constructor accepting a single SLList reference – prototype `ListVisualiser(SLList)` (Java) / `ListVisualiser(SLList *)` (C++); and
- a public method `visualise()` (Java) / `visualise()` (C++).

The `visualise()` method should display the list given in the constructor using the box-and-pointer notation you should be familiar with from lecture slides. You may use any reasonable ASCII text-based representation, for example representing the list `(1, 11, 23)` as

```
[*|*]->[*|*]->[*|/]
 |       |       |
 1       11      23
```

or at your own risk you can implement a more advanced visualisation, for example using Unicode box-drawing characters, a graphical library, or output to HTML.

You may find it helpful to test your visualiser using your own `SLList` implementation and my provided test file, using `make test`. You may also wish to consider other cases not covered by my test file, and extend my provided test file to cover those.

2.2 Peer-assessment

You must submit your `ListVisualiser` source code, along with any auxiliary files it needs to build (*e.g.* a header file) but **not** your `SLList` implementation, to the List Visualiser Activity on learn.gold by **16:00** on **Friday 30th November**.

You should include in your submission, which can be a ZIP file, any instructions for compilation or use of your code which would be helpful to another member of the class, as they will need to be able to compile and use your code in order to assess it and provide you with feedback.

Failure to submit your source code will lead to an **automatic mark of zero** for both the submission **and** the assessment portion of this activity. There **no extensions** to the deadline of 16:00 on Friday 30th November.