Data Structures (CS 102), Professor Yap – Fall 2019

HOMEWORK

October 30, 2019

Homework 5: Due on Wed Nov 6, 2019

- Carefully read the Homework Policy in the class wiki. Pay attention to the rules for submitting programs in a zip file (not rar or other formats).

PART A: WRITTEN ASSIGNMENT

A.1 (4 Points)
Question R-3.6, page 145 of Text.

How to find the second-to-last of a list.

Please write complete java code for your method with this header:

```
Node secondLast(Node u);
```

In case the list has size 0 or 1, return null.

A.2 (6 Points)
Question R-3.8, page 145 of Text.

How to find the middle of a doubly-linked list by "link hopping".

For this (and the following problem) you can write in "pseudo-code" (or in Java code which might not compile).

A.3 (8 Points)
Question C-6.21, page 254 of Text.

Non-recursive algorithm to enumerate all permutations of the set $S_n = \{1, \ldots, n\}$.

E.g., if $n = 3$, then the permutations are

$$123, 132, 231, 213, 312, 321.$$

NOTE: You may use two stacks (but no other data structure).

HINT: if one stack hold all the permutations of S_{n-1}, how do you produce another stack with all the permutations of S_n?

A.4 (6+6 Points)

Let u_0 be a node. Normally, it represents a list of size n given by

$$(u_0, u_1, \ldots, u_{n-1}, u_n)$$

where

$$u_i = \begin{cases} u & \text{if } i = 0, \\ u_{i-1}.next & \text{if } i > 0, \end{cases}$$

and u_n = null. But u_0 might also represent[1] a special kind infinite list which we call a **rho-list** because its shape looks like the Greek letter ρ named "rho"). This happens when $u_n = u_m$ (for some $0 \leq m < n$). Then the first part $(u_0, \ldots, u_m)$ is called the **tail**, and the remaining part $(u_{m+1}, \ldots, u_n)$ is called the **loop** of the rho-list. Observe that the loop is a circular list! The **tail size** and **rho-list size** are $m+1$ and n, respectively. The **loop size** is $n - m$.

(a) Write a Java method

`Node genRhoList(int m, int n)` which returns a rho-list of size n, with tail size $m + 1$, assuming that $m < n$. HOWEVER, if $m \geq n$, it returns an ordinary list of size n. Assume that your `Node` class is defined in the homework file, `hw5_Yap/src/Node.java` with members:

```
class Node {
   int val; Node next; }
```

The values inside each node are randomly generated by the node class by calling constructor with no arguments: `new Node()`.

(b) Write a method

`int sizeRhoList(Node u)` to return a int k such that the size of the list or rho-list represented by u is between k and $2k$. We have explained this method in class.

REMARK: you must put your code into the Progromming Problem B.2 below. We will grade it there.

PART B: PROGRAMMING (60 Points)

Note that we have two programs to write! You may use the targets t1 (for CircList), t2 (for RhoList) in our Makefile to test your programs. Pease do not change these targets – you may write your own variations if you like.

B.1 We have provided the file `CircList.java` that contains a dummy implementation of the following interface:

```
interface CircListInterface {
    void add(int v);     // adds a new node after head
    int remove();     // removes the node after head
    int size();       // size of the CircList
    void show(String msg);   // shows the CircList
    void rot(int n);     // rotate n times the head
    void rev();       // reverse all pointers in CircList
 }
```

Please fill in the missing methods in this file. Note that the implementation is based on a simple `Node` class which we also provide. All the randomness you need comes from calling the constructor `new Node()` (without arguments). DO NOT modify `Node.java` file or modify the `main method` in `CircList.java`.

The output from your `CircList`, using its default arguments (`ss=111, nn=8`), should look pretty close to the contents of the file `OUTPUT-CircList.txt`. The target `test1` in our Makefile should produce such an output on the terminal.

[1] But it is a special kind of infinite list, one that has a finite size (through repetition of the nodes).

B.2 Write a Java program `RhoList.java` to test your methods in the written question.

You must include your own `main` method that takes the command line arguments `ss, nn, mm` with the usual meaning. You can add other arguments if you like. Please use the default values of `ss=111`, `nn=10`, `mm=4`. The idea is to create a rho-list of size `nn` with loop size `nn-mm`.

As usually, "show" the output so that we can visually check the correctness of your code. We will grade you on the quality of this output!