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

HOMEWORK

September 28, 2019

Homework 3: Due on Friday Oct 4, 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 (5=1+2+2 Points) Consider the following code fragment. Note that the first 3 lines are taken from hw2, and they hark back to hw1:

```
Rectangle rr = new LocatedRect(1,2,3,4);
// System.out.printf(rr.top()); // ERROR!!
System.out.printf(rr.toString());
if (rr instanceof Rectangle) System.out.println("I am a Rectangle!");
else System.out.println("I am no Rectangle!");
if (rr instanceof LocatedRect) System.out.println("I am a LocatedRect!");
else System.out.println("I am no LocatedRect!");
```

Please say what will be printed by this code fragment.

A.2 (12 Points) Under folder `hw3_Yap/src`, there are two designs for a linked list: `List.java` and `NodeAsList.java`.

(a) Please describe in words the "List" design.

(b) Please describe in words the "Node-as-List" design.

(c) Please list the pros of the "List" when compared to the "Node-as-List" design. Itemize your pros and try to be concise.

(d) Please list the cons of the "List" when compared to the "Node-as-List" design.

A.3 (8 Points) In the folder, we have a third design `SList.java`: it is similar to the "List" design, but we now "sentinels" that guard both ends of the list.

(a) Please describe the design of this class.

(b) Please discuss the pros and cons (compared to "List" design)

A.4 (4 Points) Java String Patterns
Read up on pattern matching in Java. E.g., our Programming Help Page of the Class wiki (click →JavaLinks, →Java FAQs, →12. Java Strings and Pattern Matching). Which of the following words match the pattern "[^a].[bc]+"?

(1) abc

(2) 1bbbbbbbb

(3) abbbbbbbb

(4) 1zc

(5) abcbcbcbc

(6) 1c

(7) asccbbbbcbcccc

Remember to give a brief (1 sentence) justification for each answer.

A.5 (12 Points) The MatchAB Problem
The "MatchAB" problem is the following: *given two lists of strings `AA` and `BB`, to find a "solution" to these lists.* Relative to `AA, BB`, a Java pattern `Pat` can be classified as follows:

- Call `Pat` a **false negative** if `Pat` is not matched by some string in `AA`
- Call `Pat` a **false positive**. if `Pat` is matched by some string in `BB`
- Call `Pat` a **solution** if it is neither a false negative nor a false positive. In other words, `Pat` is matched by *every* string in `AA`, and it is not matched by *any* string in `BB`.

Consider this instance of the MatchAB problem:

```
AA = ("spot", "pit", "spate", "slaptwo", "respite"),
BB = ("Pot", "part", "pt", "peat").
```

The following is clearly a solution:

```
Pat="spot|pit|spate|slaptwo|respite"
```

We call this a **trivial solution** because it is just a listing of all the strings in `AA`.

(a) Please give a *nontrivial* solution `Pat1`. Try to make the length of `Pat1` as short as you can (full credit only if the length is < 10).

(b) Please give a pattern `Pat2` that is a false negative (but not a false positive).

(c) Please give a pattern `Pat3` that is a false positive (but not a false negative).

(d) Please give a pattern `Pat4` that is both a false positive AND a false negative.

A.6 (3 Points) Does every instance of the "MatchAB Problem" have a solution? First say YES or NO, and then briefly explain.

PART B: PROGRAMMING (60 Points)

Note that we have two programs to write!

B.1 Write a class called `MatchAB` to classify input patterns. This program will involve Java pattern matching, and reading of files in Java. You should test your program by using your solution to the MatchAB instance in the Written part. Here are the requirements for your class.

(a) There is a main method that takes up to 5 command line arguments. Here is a typical call:

```
> java MatchAB 0 9 src/pp.txt src/aa.txt src/bb.txt
```

The first two arguments are the usual int's, and assigns `ss = 0` and `nn = 9`. The remaining three arguments are the relative paths to a "pattern-file", an "AA-file" and a "BB-file".

(b) The pattern-file `pp.txt` contains one or more patterns. Each pattern is contained in its own line.

(c) The AA-file `aa.txt` contains a list of strings, each in its own line. The BB-file `bb.txt` is similar.

(d) Please create the 3 files `pp.txt`, `aa.txt` and `bb.txt` and put them under the `src` folder for testing. The pattern-file contains the four patterns in your answer to the Written Part. The AA-file and BB-file contain the list of strings of our input instance in the Written Part.

(e) Remark: your program may ignore the `ss` and `nn` arguments for the current homework. But, if you like, you can use them for your own testing purposes (E.g., I use `nn=0` to make my program ask for patterns from the command line, and otherwise, my program reads from the input files as we described).

(f) OUTPUT:
First, print on the screen the strings in the AA-file and BB-file.
For each pattern in the pattern-file, first print the pattern, and then print one of the four classifications of the pattern:

Pattern is a solution

Pattern is a false negative

Pattern is a false positive

Pattern is a false negative and false positive

B.2 Rewrite the `CircularlyLinkedList` class from p. 131 of Text. The inner `Node` class is from p.126. Observe that these are generic classes; you may review generics in Section 2.5.2, p. 91.

(a) The goal of your rewrite is to simplify the original code by avoiding all "access control" specifications (private, public, protected) and to remove all setters and getters.

(b) Write a `show` method that duplicates the behavior of the `show` method of the `List` class.

(c) Study the `main` method of the `List` class: you must write a `main` method for `CircularlyLinkedList` class with exactly the same behavior. In particular, it has the command line arguments, `ss` (seed) and `nn` (size). The default values are `ss=111, nn=9`. Please look at the file `List-sample-output`: **your code must duplicate the output seen in this file.**