Data Structures (CS 102), Professor Yap – Fall 2016

HOMEWORK

October 12, 2016

Homework 4: Due on Monday Oct 18, 2016

- This homework is worth 90+5 Points. In each homework, you get the extra 5 points by following all our instructions.
- READ CAREFULLY the homework instructions and policy in `http://cs.nyu.edu/~yap/wiki/class/index.php/DataStruc/Hw-page`.

PART A: WRITTEN ASSIGNMENT (30 Points)

Question A.1 (5 Points)
Consider the program:

```
                class Myst {
1.                 static class Node⟨T⟩{
2.                         T value;
3.                         Node⟨T⟩ next;
4.                          Node(T v){ value = v;}
5.                 }
                    public static void main(String[] args){
6.                         Node⟨Integer⟩ u = new Node⟨Integer⟩(5);
7.                         int n = u.value;
8.                         System.out.printf("value = %d\n", n);
                   }//main
                }//class
```

Answer these questions:
(a) Does this program compile and act as it ought?
(b) There are two odd lines in this program – what are they? Please explain what is happening. HINT: read p.197 of Text.

Question A.2 (5+2 Points) Well-formed or Balanced Parenthesis Expressions
In lecture and in the Text (Chapter 3.6), we looked at the problem of checking if a given string containing several types of parentheses are well-formed, e.g., $(\ ), [\ ], \{\ \}, \langle\ \rangle$. In practice, the string may contain other characters, but these are ignored for the purposes of balancing.

(a) What if there is only one type of parenthesis? Then we can avoid using the stack! Please describe how you can solve this problem without a stack. HINT: use a counter which you can increment or decrement.

(b) Can you still avoid a stack if you have two types of parenthesis? First say YES or NO, then try to justify your answer. You might imagine trying to use two counters to see if the problem can be solved using these

counters. Indeed you are allowed to *any* fixed number of counters, say 23 counters.

Question A.3 (6+6+6 Points) Java String Patterns

In the last homework, we looked at a string **matching** a pattern (which is represented by a string, albeit with some meta characters). We will look at **replacing** those parts of a string that match some pattern.

(a) In the previous homework, we gave you these two lists of strings:

A: `pit, spot, spate, slaptwo, respite`

B: `pt, Pot, peat, part`

We wanted a pattern that will match every string in the A-list, and fail to match every string in the B-list. Here are three possible solutions:

```
pat1 = "pit|spot|spate|slaptwo|respite"

pat2 = ".*pit.*|s.*"

pat3 = "pit|[^pP].*"
```

Now, we want you to add a single string to either the A-list or the B-list (but not to both) so that these three solutions are no longer solutions.

(b) Consider the following Java statement:

```
ss.replaceAll(pat,rep);
```

where `pat` is a pattern and `rep` is a replacement string pattern (please read up our notes or your favorite Java reference). If you are trying this out in Java, be sure to import `java.util.regex.*`.

Suppose the string `ss` is a sequence of file names that are separated by one or more white spaces. Moreover, each file name matches the pattern `"[\S]+.[\S]+"` (recall the meta-character `\S` represents any non-space character). Please design `pat` and `rep` so that the above Java statement will turn the string

```
ss = "Zoombinis.java ATM.jav Hi.class"
```

into

```
ss = "Zoombinis.class ATM.jav Hi.class"
```

I.e., any file name that matches `[a-zA-Z]+.java` is replaced by a corresponding string in `[a-zA-Z]+.class`. But do not change the other file names. HINT: use groupings in the pattern and replacement string to solve this problem.

For full credit, your pattern must also change

```
ss = "Zoombinis.java ATM.jav Hi.java"
```

into

```
ss = "Zoombinis.class ATM.jav Hi.class"
```

(c) Java Strings has a useful `split` method that splits an input string into an array of "tokens" (which are just strings). See our solution to ATM or Cash.

The split occurs in substrings that matches a "delimiter" pattern `delim`. E.g., if the delimiter are white spaces, say `delim="\s+"`, and string is `ss="a bb cccdd"` then `ss.split(delim)` returns an array of three

strings: `"a", "bb"` and `"cccdd"`. Suppose we have `delim="\s+|,\s*|\.\s*"` (i.e., spaces or comma+spaces or dot+spaces). How many tokens are there when we split each of the following 4 strings?

```
"aa dddd, m, i.n"

"aa dddd, m, i.n,.,"

"aa dddd,.  m,, i.n, , .."

", .., aa dddd,.  m,, i.n, , .."
```

First state the four numbers. Then explain the rule that `split` is using to produce its tokens. Feel free to experiment more!

PART B: PROGRAMMING ASSIGNMENT (60 Points)

All needed files are found in `hw4_Yap.zip` from Piazza/Resources. They must be included in your `src` folder and should be unmodified unless we say otherwise. Our provided Makefile has two targets `run1` and `run2` that must not be modified. You may have to tweak `run3` and `run4` to work for your own examples.

Question B.1 Grading hw3 Pattern Matching Questions (20 Points)

Please help our graders! We need a program `match/Match.java` that takes an optional pattern `pat` in the command line argument such that if each of the strings in the list A is matches `pat`, and none of the strings in list B matche `pat`, it will print "NO ERROR in both lists!". Otherwise it print one of three possible messages:

"ERROR in A list only". "ERROR in B list only", "ERROR in both lists".

You may use the A-list and B-list from hw3. You should adapt the target `run3` in the provided Makefile so that you can run your match program as follows:

```
≫ make run3 "pit|s.*|^r.*"
```

will try to check to see if the given pattern `"pit|s.*|^r.*"` produces ERRORS or not, as described above. As usual, this command line argument is optional.

Note that we assume that the default A-list and B-list are hardcoded into your program. But if you provide optional second or third arguments, these will be viewed as file names for reading the A-list or B-list from a file. We have provided sample files: `afile.txt` and `bfile.txt`. But please provide additional such files for testing.

Question B.2 Evaluation of Arithmetic Expressions (35 Points)

We have provided two files `expr/ExprAbstract.java` and `expr/Expr.java`. Please read the description of the problem in the `ExprAbstract.java` file. The class `Expr` is to be an implementation of the `ExprAbstract` class. You need to modify the file `Expr.java`, but you must not modify the `main` method in there. In the provided Makefile, you will need to adapt our provided target `run4` to run your solution with the defaults encoded in the `main` program.