Computer Science 312
Principles of Programming Languages
Fall 2024
Assignment 1

Due: beginning of class, Wednesday, 9/18/24

1. [12] Give a general description of the strings generated by each regular expression (as detailed in the slides) and list three examples in addition to the shortest string(s). Make sure your description is specific enough to capture the nature of the strings.
   (a) `([01]0)*`
   (b) `[01]+1 | [01]*0`
   (c) `[^qx][a-z]{2}\?` over the alphabet `a-z` (for the examples, use real words)
   (d) `z*yz*y[yz]* | y*zy*zy*`

2. [12] Write regular expressions for the following types of strings. For metacharacters, use only those listed in the course slides.

   (a) Bit strings of length at least 1 beginning and ending with the same bit.
   (b) File names beginning with "**HW**" (in any uppercase/lowercase combination), followed by one or more digits and ending with "`.c`" or "`.C`".
   (c) Python identifiers, which must begin with a letter or underscore and can otherwise be composed of uppercase and lowercase letters, digits, and underscores. Reserved word exceptions need not be handled here.
   (d) All telephone numbers with and without dashes/parentheses, e.g., `757-555-6789`, `7575556789`, and `(757)555-6789` (no spaces). Further, neither the area code nor exchange (the two parts composed of three digits each) are allowed to begin with 0 or 1.

3. [8] In LaTex source code, we want to find certain variables with subscripts (e.g., `${q_1}$`, which is printed as q_1). The format, however, can be more flexible with one or more spaces after the first `$` and the `{`, and before the `}` and final `$` (e.g., `$  {   q_1   }  $`. Further, the variable name and subscript can be multiple letters and digits, which require subscripts to be enclosed in curly braces (e.g., `${qq_{1a}}$` printed as qq_{1a}). Note that no additional spaces need be considered inside curly braces for the subscript. Write a regular expression to find these LaTex encodings.

4. [16] Use the following EBNF grammar (with curly braces indicating 0 or more repetitions) to

   (a) develop a leftmost derivation for the *Identifier*, `L8r2day`
   (b) develop a rightmost derivation for the *Literal*, `1.618` (using *Literal* as the start symbol)
   (c) Draw a parse tree for the derivation in (a)
   (d) Draw a parse tree for the derivation in (b)

Identifier	→	*Letter* **{** *Letter* | *Digit* **}**
Letter	→	`a` | `b` | … | `z` | `A` | `B` | … | `Z`
Digit	→	`0` | `1` | `2` | `3` | `4` | `5` | `6` | `7` | `8` | `9`
Literal	→	*Integer* | *Boolean* | *Float*
Integer	→	*Digit* **{** *Digit* **}**
Boolean	→	`true` | `false`
Float	→	*Integer* `.` *Integer*

5. [16] Consider the following grammar:

$$\begin{array}{lcl} L & \rightarrow & Y\,L \mid Y \\ W & \rightarrow & \mathbf{b}\,X \mid \mathbf{a}\,Y \mid \epsilon \\ X & \rightarrow & \mathbf{a}\,W \mid \mathbf{b}\,X\,X \\ Y & \rightarrow & \mathbf{b}\,W \mid \mathbf{a}\,Y\,Y \end{array}$$

(a) Show a parse tree for the string **a b b a b**.
(b) Show that the grammar is ambiguous using the string from (a).
(c) List all the 4-character strings generated by the grammar (derivations are not required).
(d) Provide a brief description of the strings that the grammar generates, as detailed in the slides.

6. [16] Using the following grammar:

$$\begin{array}{lcl} Exp & \rightarrow & Exp\ \mathbf{+}\ Val \mid Exp\ \mathbf{*}\ Val \mid Val \\ Val & \rightarrow & \mathbf{0} \mid \mathbf{1} \mid \mathbf{2} \mid \mathbf{3} \mid \mathbf{4} \mid \mathbf{5} \mid \mathbf{6} \mid \mathbf{7} \mid \mathbf{8} \mid \mathbf{9} \mid (Exp) \end{array}$$

(a) Draw a parse tree for the following expression and compute its final value according to the precedence rules imposed by the grammar: **3 + 2 * 4 + 5**
(b) Perform the same steps listed in (a) for the expression: **2 * 6 + 5 * 4**
(c) Rewrite the expression in (a) using allowable syntax from the grammar, but without reordering the operands, so that the final computed answer corresponds to standard mathematical precedence. State the final computed result and show a parse tree for this new expression.
(d) Perform the same steps listed in (c) for the expression in (b).

7. [20] List all of the tokens generated by scanning and parsing the following Python function. For each token, state its category: identifier, keyword, literal, operator, or delimiter. What does this function do?

```
def fun (input) :

   if not input :
      return 0

   else :
      return int (input % 10) + fun (int (input / 10))
```