Homework 1 (Posted 1/18, Due 1/25, 11:59pm, on Gradescope)

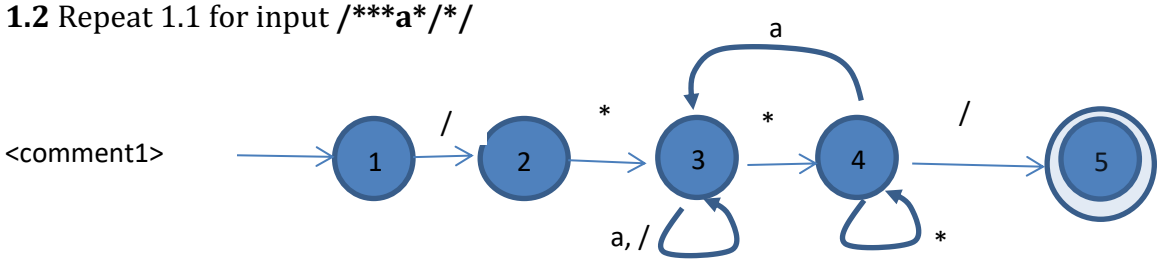
IMPORTANT NOTE: Homework solutions must be submitted on Gradescope and must satisfy the following requirements.

1. Homework solutions must be typed except for graphs that are not easy to draw, in which case hand-drawn graphs can be scanned to digital form and inserted to the submitted document.
2. Students are responsible for outlining the solution area when submitting the homework on Gradescope. **Please watch this video about submission of homework on Gradescope** (https://www.youtube.com/embed/KMPoby5g_nE?feature=oembed), especially the part about re-submission and the part on how to select pages for questions so that it's easier to grade (0'50" of the video). **Grossly unformatted pages (not properly matching questions) may be subject to a minor penalty**.
3. When composing regular expressions, use **regular definitions** for subexpressions that are otherwise longer than three operations (including concatenation, selection ,and repetition) **and** that appear more than once, e.g., <digit> = [0-9] and <letter>=[a-zA-Z]. Moreover, if a regular expression has a long sequence of mixed operations (mixing concatenation, selection, or Kleene closure), please also introduce **regular definitions** to separate logically meaningful subexpressions for better readability. Such regular definitions are then used to compose the final regular expression, as shown in lectures. If a regular expression's composition is still complex after introducing regular definitions, please give explanation why the presented regular expression satisfies the requirement
4. The following shortcut notations are permitted when composing regular expressions: R+, which means one or more appearances of R; R?, which means zero or one appearance of R; and range notation such as [1-7]. All other shortcut notations (not mentioned in the lectures), unless provided in the problem statement, must be explained before used in the submitted solution.
5. When we seek a finite automaton or a regular expression to accept a specified set of input, it is implied that nothing else should be accepted. Please strictly follow the format in the lectures for drawing finite automata in the lectures. Do not use formats seen elsewhere, such as those seen on the internet.

Problem 1 (20 pts)

1.1 Refer to the following finite automaton. Is input **/*aa/***/** accepted by the DFA? If not in which state will the error be detected and what is the error? If it is accepted, sequentially list all the states that are visited (with possible revisits) under the given input.

1.2 Repeat 1.1 for input **/***a*/*/**



Problem 2 (15 pts)

Draw a finite automaton that accepts all finite and non-null sequences of 0's, 1's and 2's such that the same digit never appears next to each other. Use no more than four states in the finite automaton.

Problem 3 (15 Pts)

Write a regular expression to match all multi-line comments as in Java. Such a comment is a comment body quoted in a single pair of "**/***" and "***/**". The characters in the comment body are assumed to be the following:

- Characters / and *. To avoid confusion, these characters and their combinations should be double quoted in the regular expression
- All characters that are neither / nor *. To simplify the writing of the regular expression, use [^/] to denote any character, including *, that is not /, use [^*] to denote any character, including /, that is not *, and use [^*/] to denote any character that is neither / nor *.

You are reminded that the comment body must contain no substring "*/". Please use regular definitions to define subexpressions such that the composition is easier to understand, as we required in the guideline given on the first page.

Problem 4 (20 pts).

The following grammar has **<prog>** as its starting nonterminal. The numbers are used to identify individual rules.

1. S → ID = E ;
2. S → write (ID) ;
3. E → read

4′. E → F E′

5′. E′ → ε

5″. E′ → + F

6. F → ID
7. F → num

8′. L → S L′

9′. L′ → ε

9″ L′ → L

10. <prog> → L

Write a complete *left-most* derivation steps for input "**x = a+2; write (y);**" (Note: refer to *x, y, a* as IDs in the derivations. Refer to the integer value 2 as **num** in the derivations. If the derivation steps cannot continue before the input is exhausted, please point out what the remaining input is.)

Problem 5 (15 pts) The following grammar is ambiguous.

A → − A
| A − id
| id

Show two leftmost derivations for input – id - id

Problem 6 (15 pts) Is the following grammar ambiguous? If it is, please give an example input and show two different leftmost derivations. Make the example input as short as you can.

1. S → Stmt
2. Stmt → IfStmt
3. Stmt → OtherStmt
4. IfStmt → if (Expr) Stmt
5. IfStmt → if (Expr) Stmt else Stmt